MASTER IN COMPUTER SCIENCE

A multimedia support to the learning of interaction objects

Bodart, Thibaut; Magnier, Marc Laurent

Award date:
1999

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

A Multimedia Support to the Learning of Interaction Objects

Thibaut Bodart
Marc-Laurent Magnier

Thesis submitted in fulfilment of the requirement
for the degree of Master in Computer Science

-September 1999-

Supervisor: Prof. François Bodart

RUE GRANDGAGNAGE, 21 • B-5000 NAMUR (BELGIUM)

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Un Support Multimédia à l'Apprentissage des Objets Interactifs

Thibaut Bodart
Marc-Laurent Magnier

Mémoire présenté en vue de l'obtention
du diplôme de Maître en Informatique

-Septembre 1999-

Promoteur : Prof. François Bodart

RUE GRANDGAGNAGE, 21 • B-5000 NAMUR (BELGIQUE)

Abstract

This thesis describes the development of a multimedia environment for the learning of Interaction Objects (IOs). This work is part of the VESALE project of the Institut d'informatique and follows a five month internship period at the University of Port-Elizabeth (South Africa).

The objective of this environment is to give the student a good comprehension of the IO concept as well as a method to select them.

The main challenge is to adapt the paper course for the web environment used in the project. This adaptation requires an entire rethinking of the way of presenting a course. The multimedia environment at our disposal gives us the following opportunities to create additional values for the course:

- Interactive applications
- Increased freedom in accessing information
- The ability to organise the course in different pedagogical scenarios.

Résumé

Ce mémoire décrit le développement d'un environnement multimédia pour l'apprentissage des Objets Interactifs (OI). Ce travail s'inscrit dans le cadre du projet VESALE de l'institut d'informatique et fait suite à un stage de cinq mois à l'Université de Port-Elizabeth (Afrique du Sud).

L'objectif de cet environnement est de permettre à l'étudiant d'acquérir une bonne compréhension du concept d'OI, ainsi qu'une méthode pour les sélectionner.

Le défi principal est l'adaptation du cours papier à l'environnement web utilisé pour le projet. Cette adaptation nécessite de repenser complètement la façon de présenter un cours. En effet, l'environnement multimédia à notre disposition nous donne les opportunités suivantes d'ajouter une plus-value au cours :

- Des applications interactives
- Une plus grande liberté dans l'accès à l'information.
- La possibilité d'organiser le cours en plusieurs scénarios pédagogiques.

Acknowledgements

Nos remerciements les plus sincères s'adressent

Au Professeur François Bodart, promoteur de ce mémoire, non pas tant pour la confiance qu'il nous a témoigné depuis le début de ce travail que pour son intérêt, son attention et son appréciation progressivement croissante au cours de son évolution. Nous tenons également à le remercier pour tout les conseils qu'il nous a donné au cours de la réalisation de ce travail.

To Professor Janet L. Wesson, University of Port-Elizabeth, for her friendly reception, for the attention she gave us during our stay at UPE and for accepting to be a member of our jury.

To Mr and Mrs Dekock, University of Port-Elizabeth, for their friendliness and for showing us how beautiful South Africa is.

To Lean Nicholls, University of Port-Elizabeth, for his help on our project.

To the rest of UPE staff.

A l'équipe VESALE, à savoir Jean-Marie Leheureux, Efrem M'baki et Abdo Bereikdar pour leurs apports aux réunions du projet.

A Anne-Sandrine « vicky » Rosmant, pour ses précieuses notes de cours, sans lesquelles nos études se seraient terminées en deuxième candi.

A Rudy « bloub » Michiels et Xavier « cucumber » Gillmann pour les parties endiablées de Starcraft qui nous ont permis de nous détendre pendant ces longs mois de juillet et août.

A tous les étudiants de 3ème maîtrise, pour ces années passées ensemble.

Nos derniers remerciements, et non des moindres, nous les destinons à nos parents sans qui rien ne serait. Ils n'ont jamais cessé d'être des parents attentifs et de montrer leur disponibilité en toutes circonstances. Nous voudrions tout particulièrement les remercier pour nous avoir donné l'opportunité de nous construire un avenir. Et c'est tout naturellement, en reconnaissance de tout cela, et de biens d'autres choses encore, que nous voudrions faire un peu leur le travail qui fût le notre en leur dédicaçant bien volontiers l'ensemble de ce mémoire.

Table of content

Chapter 6. Technology choices

Chapter 7. Design

1

Introduction

1. Subject

The subject of this thesis is the development of the part of the VESALE project[1] concerning the Interaction Objects (IOs). The objective is to give the student a good comprehension of the IO concept as well as a method to select them.

The main challenge is to adapt the paper course for the web environment used in the project. This adaptation requires an entire rethinking of the way of presenting a course. The multimedia environment at our disposal gives us the following opportunities to create additional values for the course:

- Interactive applications
- Increased freedom in accessing information
- The ability to organise the course in different pedagogical scenarios.

As we are students in computer science and not pedagogues, the pedagogical aspect of this work is primarily based on our personal experience. As far as pedagogy is concerned, the course we will develop must be regarded as a prototype that will have to be criticised by pedagogues. Nevertheless, some personal research in that field appears to be essential in order to provide some consistency.

[1] See chapter 2

2. Internship at the University of Port Elizabeth (South Africa)

From September to January, we worked at the University of Port Elizabeth (UPE) in South Africa. Our supervisor was Professor Janet L. Wesson.

During this internship period, we developed a library of IOs in Java. This project was realised as part of a BSc Honour degree. The title of the report we had to write in that occasion is "*The Representation of Abstract Interaction Objects for Reusable Object Design*". You will find it in Appendix A.

The main things that we got out of this experience are a good knowledge of the Java programming language as well as a deeper understanding of the IOs functioning. Moreover, the supervision that we received during the writing process of the report was very useful for this thesis redaction.

On a personal point of view, this immersion in such a friendly and interesting environment was an unforgettable experience.

3. Structure of this thesis

The structure of this thesis is the following:

In *chapter 2*, we will shortly describe the VESALE project and more precisely which part of it we will develop.

In *chapter 3*, we will gather some pedagogical knowledge that can apply to multimedia environments.

In *chapter 4*, we will research some design guidelines for the development of a web-based learning environment.

In *chapter 5*, we will go trough the analysis process. We will describe the expected features of our environment.

In *chapter 6*, we will research the most suitable technologies to implement the desired environment.

In *chapter 7*, we will use those technologies to implement the environment.

Finally, in *chapter 8*, we will evaluate our environment in terms of pedagogical contribution to the learning of IOs.

2

The VESALE project

1. Introduction

Teaching and research related to Human-Computer Interface (HCI) have as aim the theories, the models, the methods and the tools necessary to all the stages of HCI life cycle [Beirekdar99]. Consequently, would it be useful to use the specific knowledge from this field in the development of software tools supporting computer-aided teaching? Moreover, if this teaching is in the field itself, the opportunity is given to use its content in self-illustration, therefore enriching the learning. This principle of *bootstrapping* is fundamental to the VESALE project (Visual user interface design Education Supported by a computer-Aided Learning Environment), a multimedia environment supporting the teaching of HCI.

This teaching's support concerns four situations:

- The apprenticeship as part of the live teaching: teacher and students are face to face
- The complementary apprenticeship to the live teaching which essentially stands the evaluation of knowledge and review of syllabus contents
- The indirect teaching or distant learning, i.e. the self-apprenticeship enriched by the interactions with the teacher
- The co-operation to the teaching's enrichments. In particularly, the enrichments for reasoned cases and for illustrations of multimedia technologies.

2. Global presentation of the VESALE project modules

The multimedia environment chosen to develop the VESALE project is the World Wide Web.

This project is composed of several modules, each of them being linked to the others. The modules are the following [Beirekdar99]:

- **The course notes module**

The course notes module contains the notes of the HCI course presented in the form of web pages.

- **The reasoned cases module**

The reasoned cases module contains criticised examples of interfaces. This base can be enriched by cases suggested by students or other interested persons. Rudy Michiels and Gaëtan Prévot have tackled this subject [Michiels-Prevot99].

- **The multimedia technologies illustration module**

The multimedia technologies illustration module aims to favour the illustration, the explanation and the comprehension of the multimedia interaction techniques

- **The ergonomic rules module**

The ergonomic rules module is intended to illustrate the use of the ergonomic criteria and rules in order to build useful and usable interfaces.

- **The knowledge evaluation module**

The knowledge evaluation module contains exercises and questions allowing a summative and formative evaluation.

- **The video sequences module**

The video sequences module contains video sequences illustrating some specific parts of the course.

- **The dialogue space**

The dialogue space supports the dialogue between the professor and the student.

- **The comments space**

The comments space is associated with each particular base. This space makes it possible for the students to express comments as well as for the professor to react.

The figure 2-1 describes the relationships between the modules.

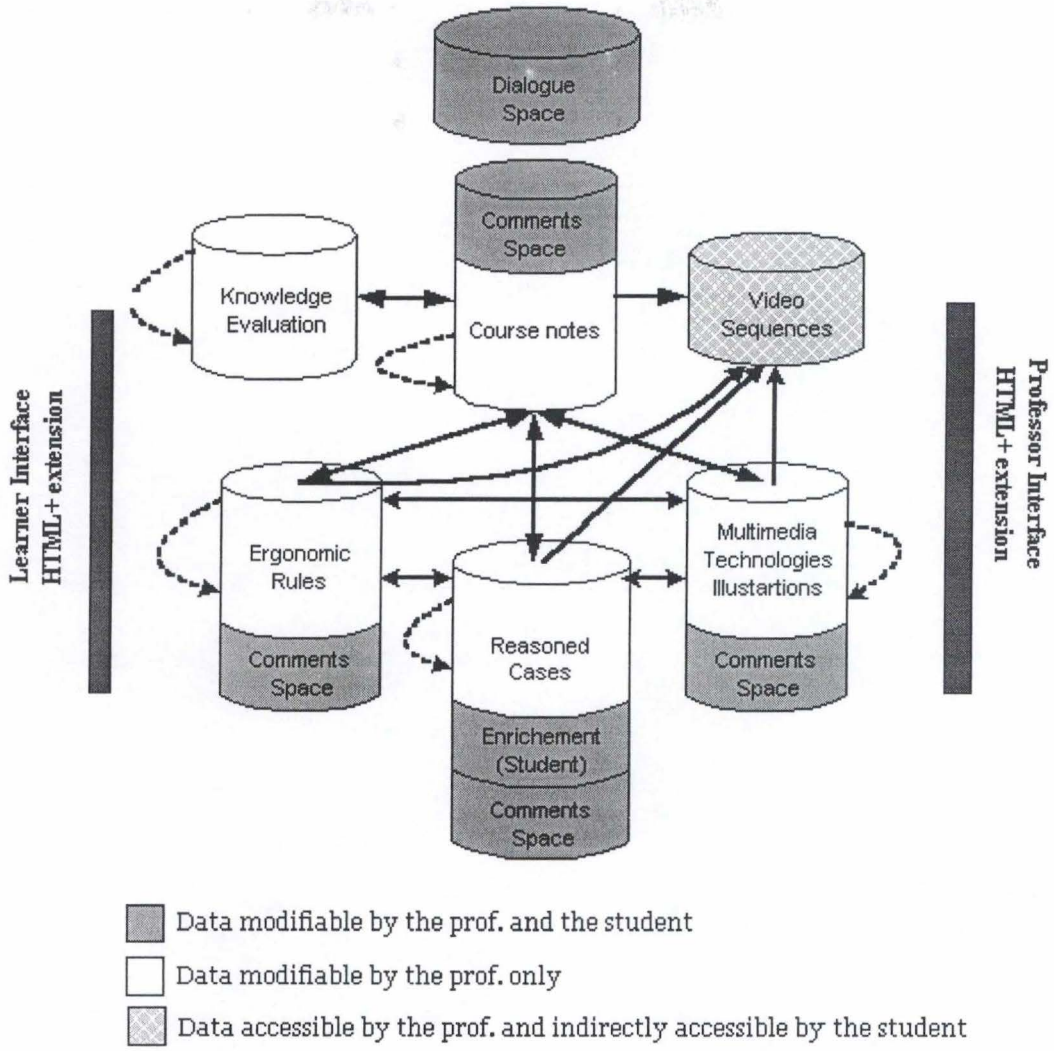


Figure 2-1. *The relations between the different modules of the VESALE architecture*

3. Our part of the VESALE project

Our part of the VESALE project consists of the development of the multimedia technologies illustrations base along with the course notes related to the IOs. The development of the comment spaces attached are not of our concern.

3.1. The multimedia technologies illustrations base

This base aims to favour the illustration, the explanation and the comprehension of the course concepts, of the multimedia interaction techniques by using the concepts and the multimedia technologies themselves in auto-illustration [Vanderdonckt98]. This illustrations base could be composed of screen shots, of representatives photos, of process graphical animations, of video sequences illustrating some actions involving the human being, of synthetic vocal comments, of simulations of interactive sessions about different software systems, of real interactive applications with session recording, of interactive applications implemented according to certain measures.

3.2. Course notes

We also have to develop the course related to IOs. These notes will be presented to the students in the form of a hypermedia syllabus.

3

Web-Based Learning

1. Introduction

When placing their courses on the Internet, for either distance education or as extensions of traditional classroom teaching methods, designers should not rush. As a matter of fact, using new media requires new approaches to teaching. Putting course content online is more than a matter of converting the syllabus to HTML and placing it on a server. There are important considerations on what should and shouldn't be placed online and what tools work best to reach an instructional goal.

As computer scientists, our approach is primarily a technical one, centred on the implementation of technologies that could be used in a pedagogical environment. We have tackled this problem as computer scientists and not as pedagogues. The result of this work should therefore be regarded as a **prototype** that will have to be criticised by pedagogues.

Nevertheless, some pedagogical concern must guide us in our approach while developing this environment. First, we will confront two opposite approaches (Constructivism and Objectivism). Then, we will present six teaching/apprenticeship paradigms. For those that are relevant to our project, we will define which applications we could develop in order to implement them. Finally, we will suggest two ways of structuring the course.

2. Constructivism versus Objectivism

Historically, teachers have used *Objectivist* methods in which students are presented information that they repeat back to the teacher. The underlying model of Objectivism is Behavioral Psychology (based on B.F. Skinner's work[1]). Behaviourists view psychology in terms of resulting behaviours which can be modified by consequences (rewards and punishments).

The current trend in education appears to be Constructivism which is based on Cognitive Psychology. Under this model, students are viewed as active processors of information [Koyanagi99].

The table 3-1 compares both opposite learning approaches.

Objectivism	**Constructivism**
Method: • Content presentation • Question is put to student • Student is told if answer is right • Positive reinforcement for right answers • Cycle is repeated for wrong answers	Methods vary: • Encourage knowledge formation • Process is different for each student • Self-directed exploration • Discovery learning • Construction of concepts, schema and mental mode
External truths and knowledge exists for learners to memorise	Truth and knowledge is constructed by students based on perspective and experience
Teacher controls	Teacher observes, coaches and facilitates
Students learn meaning	Students create meaning

***Table 3-1**. Objectivism versus Constructivism in education*

While Skinnerian methods have been effective in learning how to train animals and helping human beings modify their behaviour, the behaviourists fell short of what is most important in education. Education is more than just modifying behaviours. It is also about helping the student to learn how to develop strategies for learning. Such is the goal of the cognitive movement in education as defined by Bruning[2].

As articulated by Piaget[3], students learn better when they can invent knowledge through inquiry and experimentation instead of acquiring facts presented by a teacher in class. It is difficult for a teacher to provide this kind of environment for each student in a traditional classroom. Since there is only one teacher for many students, it is physically impossible for the teacher to support each student's individual needs. On the other hand, multimedia computers provide a powerful environment for helping achieve the goals of the cognitive movement in education.

[1] Skinner, B. F. The Behavior of Organisms. New York: Appleton-Century-Crofts, 1938.
Skinner, B. F. Science and Human Behavior. New York: Macmillan, 1953.
[2] Bruning, Roger H., Schraw, G. J., and R. R. Ronning. Cognitive Psychology and Instruction. Englewood Cliffs, N.J.: Prentice Hall, 1995.
[3] Piaget, J. The Mechanisms of Perception. New York: Basic Books, 1969

3. Teaching/Apprenticeship paradigms

Dieudonné Leclercq and Brigitte Denis [Leclercq98] describe six teaching/apprenticeship paradigms. Three of them are under the control of the teacher and the others are under the control of the learner as shown in figure 3-1.

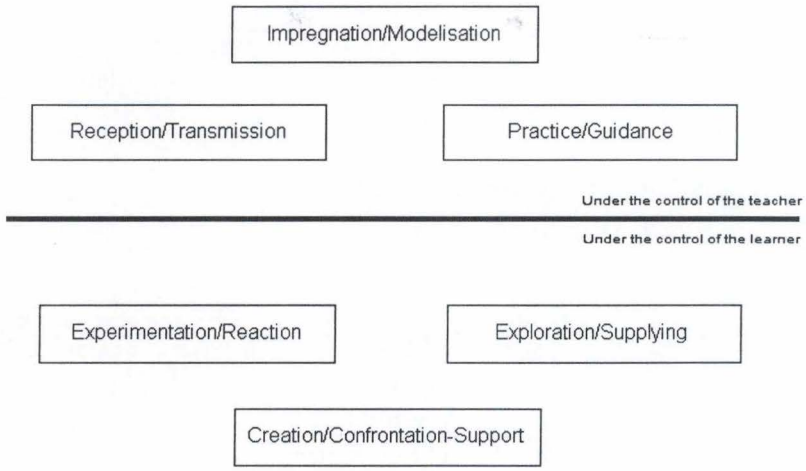


***Figure 3-1**. The six teaching/apprenticeship paradigms*

3.1. Paradigms under the control of the teacher

3.1.1. Impregnation / Modelisation

Paradigm 1: Impregnation/Modelisation	
Learners operation	Impregnation
Teacher operation	Model builder
The learner says	"Show me"
The teacher says	"Look at me doing it"
Example	Students in art history learn a lot from going to the Acropolis in addition to their theoretical knowledge about it.

3.1.2. Reception / Transmission

Paradigm 2: Reception / Transmission	
Learners operation	Reception
Teacher operation	Transmission
The learner says	"Tell me"
The teacher says	"You must know that…"
Example	A traditional university lecture.

3.1.3. Practice/Guidance

Paradigm 3: Practice/Guidance	
Learners operation	Practice
Teacher operation	Guidance
The learner says	"Correct me"
The teacher says	"Here is your mistake"
Example	A coach corrects his athlete when he is doing a mistake.

3.2. Paradigms under the control of the learner

3.2.1. Exploration/Supplying

Paradigm 4: Exploration/Supplying	
Learners operation	Exploration
Teacher operation	Supplying
The learner says	"Let me explore"
The teacher says	"Here is what is available"
Example	A library gives the student the opportunity to look for information about a particular subject.

3.2.2. Experimentation/Reaction

Paradigm 5: Experimentation/Reaction	
Learners operation	Experimentation
Teacher operation	Reaction
The learner says	"Let me manipulate"
The teacher says	"Here are the cases and the resources "
Example	A flight simulator

3.2.3. Creation/Confrontation-Support

Paradigm 6: Creation/Confrontation-Support	
Learners operation	Creation
Teacher operation	Confrontation-Support
The learner says	"Let me build"
The teacher says	"I support you ", "Here is what I think about it"
Example	A writer learns by creating a novel and then confronts it to other people's judgement.

3.3. Formation strategy

Most formation systems combine several paradigms in order to compensate the weaknesses of some with the strength of others. It is indeed not common to find a teaching/apprenticeship situation based on only one paradigm. The combination of several paradigms makes a *formation strategy*.

As a reminder, the VESALE environment will be used in the following situations[4]:

- The apprenticeship as part of the live teaching
- The complementary apprenticeship to the live teaching
- The indirect teaching or distant learning
- The co-operation to the teaching's enrichment

Considering those situations as well as the specific part of the VESALE environment that we have to develop, we can consider the following paradigms:

– **Paradigm 1: Impregnation / Modelisation**

This paradigm could be applied as part of the face to face interaction between the students and the teacher. For instance, we could imagine a teacher illustrating concepts like Interaction Objects (IOs) in front of the students by manipulating them in a multimedia environment.

– **Paradigm 2: Reception / Transmission**

An on-line version of the syllabus concerning the multimedia technologies could be provided to the students allowing them either to review the course or to learn it distantly.

– **Paradigm 3: Practice/ Guidance**

Our environment could provide applications allowing the students to manipulate concepts such as selection trees. The environment could play the role of the guide by helping the students in their manipulations.

– **Paradigm 4: Exploration / Supplying**

Our environment could allow the students to navigate freely inside the multimedia technologies base.

– **Paradigm 5: Experimentation / Reaction**

Our environment could provide applications allowing the students to manipulate concepts. For instance, this environment could allow him to experience the manipulation of IOs, displaying immediately the result of his actions.

[4] See chapter 2 section 1

- **Paradigm 6: Creation / Confrontation, Support**

One could wonder if the programming of IOs by students is a good pedagogical approach in their teaching. We followed this method during our training period at the University of Port-Elizabeth and found that it was a very good way of understanding the behaviour of IOs. Nevertheless this approach would not fit well into this part of the VESALE project.

Therefore, our formation system could combine five paradigms. However, even if we will keep all five paradigms in mind while developing this environment, our focus will be set essentially on two of them, namely: the *practice/guidance* and *experimentation/reaction* paradigms. It is indeed in the application of those two paradigms that the learning process can be improved the most by multimedia technologies.

4. Pedagogical scenarios

In addition to the theoretical concepts, and as stated in our formation strategy (3.2), the course could contain practical applications. Considering those two complementary parts, we could think of two different approaches for the course presentation. We have decided to call these approaches *pedagogical scenarios*. According to their underlying philosophy, we have chosen to call them the *Objectivist* and *Constructivist* scenarios.

4.1. The Objectivist scenario

In the Objectivist scenario, the course concepts are presented initially to the student. When the student has read everything he has to know about a particular subject, applications that summarises all the concepts related are presented to him.

At this stage, he could manipulate applications in order to **confirm** that he has a good understanding of the concepts. Naturally, the definitions of every concepts used in the applications are accessible to the student. Those accessing capabilities are working here as a *reminder* as shown in figure 3-2.

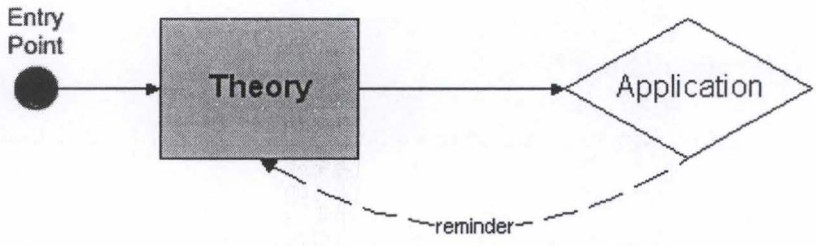


Figure 3-2. In the Objectivist scenario, the entry point is the theory.

4.2. The Constructivist scenario

One can wonder if the accessing capabilities of applications could not be used to reverse the problem. As a matter of fact, presenting applications first and allowing the student to interact with concepts that he doesn't already master corresponds to a Constructivist approach.

In the Constructivist scenario, the student interacts initially with applications. As soon as he is confronted with concepts that he doesn't understand, he can refer to the theory related as shown in figure 3-3. Thanks to this method, the student can create his own mental model of the subject and therefore becomes an active processor of information.

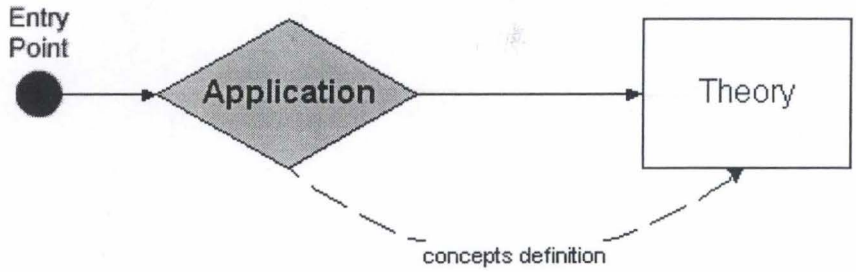


***Figure 3-3**. In the Constructivist scenario, the entry point is the application.*

4.3. Scenarios

In order to develop this learning environment we have to decide which scenario we will implement.

An *Objectivist* scenario would follow almost the same structure as the paper course but will create additional value in the way of interactive applications allowing the users to review the concepts. On the other hand, a *Constructivist* scenario would allow the students to apprehend the concepts via the interactive applications.

As our objective is to develop a prototype, we can not exclude at the very outset any scenarios. Therefore, we will implement both scenarios and will try to evaluate ultimately their respective contributions to the learning of IOs.

5. Conclusion

In order to develop our part of the project, we will try to implement two distinct scenarios keeping the same ultimate objective in mind: making the student an active processor of information (even in the Objectivist scenario (4.1)). The web environment gives us lots of opportunities to fully achieve this goal.

4

Guidelines for the design of a multimedia course

1. Introduction

As seen in Chapter 3, the pedagogical scenarios suggested will contain course notes as well as interactive applications which will be implemented using web technologies. Therefore, it is necessary to research some guidelines that we help us in our developing process.

First, we will describe some general web design guidelines and adapt them for the context of an online course. Secondly, we will analyse some existing pedagogical interactive applications in order to deduce design principles.

2. Web-design guidelines for a multimedia course

In this section we will start by defining the key concepts of utility and usability. Then we will examine the different ways of organising information. Afterwards we will describe how to write it for the web. Finally, we will investigate the best ways of displaying this information.

The following guidelines apply mainly to the development of web sites. Nevertheless, they almost all identically apply to online courses. However, we will adapt them when needed.

2.1. Utility and Usability

When designing a web site, one of the most important things to keep in mind is improving its usability. This concept of usability has been fully described by Jakob Nielsen [Nielsen98a]. He gives the following definition: "*Usability* is the measure of the quality of the user experience when interacting with something – whether a web site, a traditional software application, or any other device the user can operate in some way or another"

The usefulness of a system is determined by two components:

- *Utility*: Does the system do anything that people care about? If the system does something irrelevant or if it doesn't solve the main problem, then it does not matter whether it is easy to use: it will be a poor system in any case.
- *Usability*: Can the user use the system and can he or she do so effectively? Even if the system does exactly the right thing in theory, it will still be a poor system if the user cannot figure out how to get it to work.

Usability is not a single number but has five characteristics:

- *Ease of learning*: How fast can a user, who has never seen the user interface before, learn it sufficiently well to accomplish basic tasks?
- *Efficiency of use*: Once an experienced user has learned to use the system, how fast can he or she accomplish tasks?
- *Memorability*: If a user has used the system at some earlier date, can he or she remember enough to use it more effectively next time (or does the user have to start over again learning everything every time)?
- *Error frequency and severity*: How often do users make errors while using the system, how serious are these errors and how easy is it to recover from a user error?
- *Subjective satisfaction*: How much does the user like using the system?

All five characteristics of usability need to be considered in any design project but some of them are more important than others. For the web, ease of learning is often the most important usability attribute since users rarely spend enough time on any individual web site to become expert users who care more about efficiency. Also, subjective satisfaction is critical since users can go anywhere else on the web at he click of a mouse. User errors are less critical on most web sites, though E-commerce sites must take steps to ensure that users order the right products and enter their credit card and shipping address correctly.

In the context of a multimedia course, we think that the focus must be set on the *efficiency of use* and *memorability* rather than on the ease of learning since the same student will use the course repeatedly. The *user satisfaction* is also very important since the aim of the project is to attract the students towards the online version of the syllabus, which offers more features.

2.2. Information architecture

The *information architecture* of a web site defines how information is organised. In order to build an information architecture, we have to gather information, split it into chunks and then reorganise it.

2.2.1. Content "chunking"

The way people seek and use reference information suggests that smaller, discrete units of information are more functional and easier to handle that long, undifferentiated tracts [Lynch99]. This method for presenting information translates well to the web for several reasons:

- Discrete chunks of information lend themselves to web links. The user of a web link usually expects to find a specific unit of relevant information, not a book's worth of content.
- Chunking can help organise and present information in a uniform format. This allows users both to apply their past experience with a site to future searches and explorations and to predict how an unfamiliar section of a web site will be organised. This element is particularly important in our context where the mental model that a student can have of the course concepts depends on the organisation of its content.
- Concise chunks of information are better suited to the computer screen, which provides a limited view of long documents. Long web pages tend to disorient readers; they require users to scroll long distances and to remember what is off-screen.

The concept of *information chunk* must be flexible and consistent with common sense, logical organisation, and convenience. The nature of the content must suggest how it should be subdivided and organised. In the context of a course, we think that chunking the course according to its concepts is appropriate.

Although short web documents are usually preferable, it makes sense to provide an additional undivided version of the document in case the user want to be able to print it easily or save it in one step.

2.2.2. Organisation structures

The technology used to organise information is hypertext. It serves as a basis for structures that organise content hierarchically, sequentially or conceptually.

2.2.2.1. Hypertext

In a web site, the way people access information is hypertext. *Hypertext* is a highly non-linear way of structuring information [Rosenfeld98, p.40]. A hypertext system involves two primary types of components: the items or chunks of information which are to be linked, and the links between those chunks. These components can form hypermedia systems that connect text, data, image, video, and audio chunks. Hypertext chunks can be connected hierarchically, non-hierarchically, or both as shown in figure 4-1.

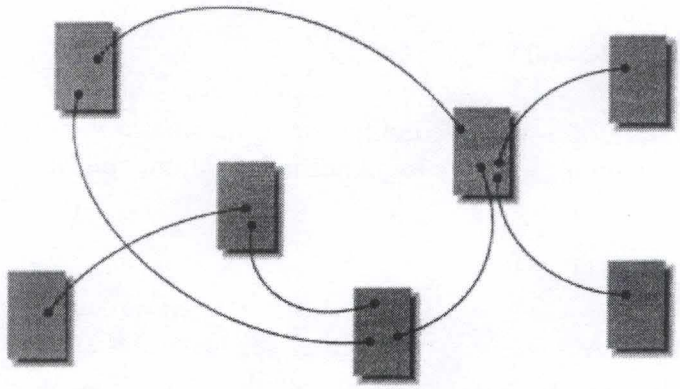


***Figure 4-1**. In hypertext systems, content chunks are connected via links in a loose web of relationships.*

Although this system provides great flexibility, it presents substantial potential for complexity and user confusion. As users navigate through highly hypertextual web sites, it is easy for them to get lost. They simply can't create a mental model of the site organization. Without context, users can quickly become overwhelmed and frustrated.

2.2.2.2. Hierarchical organisation

Information can be organised *hierarchically* [Rosenfeld98, p.38]. When designing information hierarchies on the web, it is essential to consider the balance between breadth and depth. *Breadth* refers to the number of options at each level of the hierarchy. *Depth* refers to the number of levels in the hierarchy. If a hierarchy is too narrow and deep, users have to click through an inordinate number of levels to find what they are looking for (Figure 4-2). If a hierarchy is too broad and shallow, users are faced with too many options on the main menu and are unpleasantly surprised by the lack of content once they select an option (Figure 4-3).

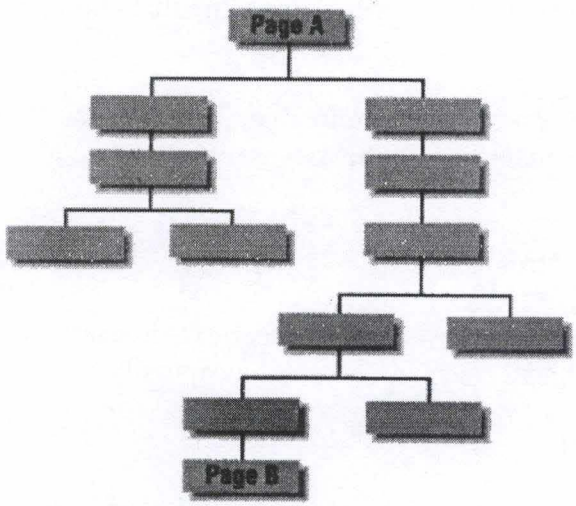


***Figure 4-2**. In this narrow and deep hierarchy, users are faced with six clicks to reach the deepest content*

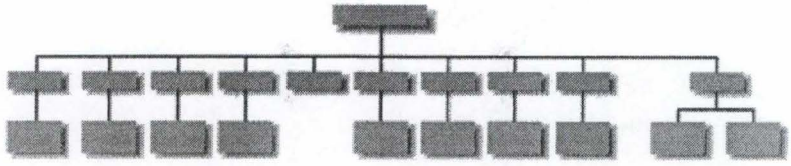


Figure 4-3. *In this broad and shallow hierarchy, users must choose from ten options to reach a limited amount of content*

In considering breadth, designers must pay attention to the cognitive limits of the human mind. With the exception of particular cases, the seven plus-or-minus two rule[1] must be followed. Web sites with more than ten options on the main menu can overwhelm users.

In considering depth, it is important to be even more conservative. If users are forced to click through more than four or five levels, they may simply give up and leave the web site. At the very least, they'll become frustrated.

2.2.2.3. Sequential organisation

Besides the hierarchical organisation, a *sequential* organisation can be provided. As students are used to this kind of organisation in their paper courses, it seems to be particularly suitable for an online course. This organisation depends on the hierarchical one and is therefore a complementary structure.

2.2.2.4. Conceptual organisation

The *conceptual* organisation is based on the relationships existing between the different information chunks. For instance an information chunk describing a particular concept could be linked to other chunks describing either prerequisites, father or child concepts.

2.3. Writing for the web

It is an unfortunate fact that current computer screens lead to a reading speed that is approximately 25% slower than reading from paper [Nielsen96b]. Better screens have been invented and it is just a matter of time before reading from computers is as good as reading from paper, but for the time being information must be designed for the actual screens in use around the world.

The reduced reading speed on computers can be compensated by good hypertext design that allows the user to read less information and to find it faster.

On the other hand, other types of information do require the user to read large amounts of text. Those users typically may prefer not having to sit at their screen while doing so. Thus, even when the reading speed problem gets solved, we may still find that people decide to print out long texts rather than read them on the screen.

---

[1] G. Miller, "The Magical Number Seven, Plus or Minus Two : Some Limits on our Capacity for Processing Information", *Psychological Review* 63, no. 2 (1956): 81-97.

2.3.1. Concise, scannable and objective writing

Usability studies conducted over the last four years by Jakob Nielsen have brought to the fore three main content oriented conclusions [Nielsen97]:

- Users do not read on the web; instead they **scan** the pages, trying to pick out a few sentences or even parts of sentences to get the information they want
- Users do not like long, scrolling pages: they prefer the text to be **short** and to the point
- Users detest anything that seems like subjective information and prefer a more **factual** writing style

From these observations he defines three properties of a good written online text: *conciseness*, *scannability* and *objectivity*.

First, texts should be *concise* because users like to get information quickly. Writing concise texts requires not only tightening of language, but also cutting of overly detailed information. A good rewritten text should have 50% of the word count of its paper equivalent.

Texts should also be *scannable* because reading from computer screens is tiring and, therefore, users attempt to minimise the number of words they read. In order to make a web page scannable, designers must try to use as much as possible the following techniques:

- Keywords highlighting
- Bulleted and numbered list. It slows down the scanning eyes and draws attention to important points
- Each paragraph should contain one main idea; a second paragraph should be used for a second idea, since users tend to skip any second point as they scan over the paragraph
- A page should start with the conclusion as well as a short summary of the remaining contents (inverted pyramid style)

Finally texts should be *objective*. The information should be presented without exaggeration, subjective claims, or boasting.

The text excerpt in the figure 4-4 is written in a promotional style.

> Nebraska is filled with internationally recognized attractions that draw large crowds of people every year, without fail. In 1996, some of the most popular places were Fort Robinson State Park (355,000 visitors), Scotts Bluff National Monument (132,166), Arbor Lodge State Historical Park & Museum (100,000), Carhenge (86,598), Stuhr Museum of the Prairie Pioneer (60,002), and Buffalo Bill Ranch State Historical Park (28,446).

Figure 4-4. *A text written in a promotional style.*

The same text, rewritten in order to be more concise, scannable and objective is shown in figure 4-5.

In 1996, six of the most-visited places in Nebraska were:

- Fort Robinson State Park
- Scotts Bluff National Monument
- Arbor Lodge State Historical Park & Museum
- Carhenge
- Stuhr Museum of the Prairie Pioneer
- Buffalo Bill Ranch State Historical Park

Figure 4-5. *The rewritten text.*

To measure the effect of some of the content guidelines he had identified, Jakob Nielsen measured the usability of these two excerpts among a group of test users. His conclusion was that the rewritten version had a 124% better usability than the original one.

2.3.2. Printing version

Readers will often want to print material from the site and read it later from paper. In order to make it more convenient, a site should provide two versions of its content: one that is optimised for online viewing and one that is optimized for printing (has good layout and is in one piece). This print file should probably be in formats like PostScript or PDF. The URL of the online version should be included within the text of the page so that users can find updates and correctly cite the source.

In a distant learning context, this printing version is particularly important because it will be used as the course notes by the students.

2.4. Page design

Users seek clarity, order, and trustworthiness in information sources, whether traditional paper documents or web pages. Effective page design can provide this confidence. The spatial organisation of graphics and text on the web page can engage readers with graphical impact, direct their attention, prioritise the information they see, and make their interactions with the web site more enjoyable and efficient [Lynch99, p.53]. Those considerations must also apply when designing a web-based course.

2.4.1. Site identity

Superficial though it may seem, users enjoy some sites simply because they are aesthetically pleasing [Lynch99, p.56]. However, it is rarely because they simply contain the most pleasing graphics. An attractive site is distinguished by a cohesive and consistent look that presents a *unique identity*. These sites graphics and page layouts are integrated with their other features such as navigation systems, custom applications, editorial style, and so forth. Therefore, the user doesn't notice the individual images so much as he or she enjoys the overall atmosphere and experience created by the site.

Designers must establish a layout grid and a style for handling text and graphics, then apply it consistently to build rhythm and unity across the pages of the site. Repetition is not boring; it

gives the site a consistent graphic identity that creates and then reinforces a distinct sense of place and makes the site distinct and memorable (Figure 4-6). A consistent approach to layout and navigation allows readers to adapt quickly to the design and predict with confidence the location of information and navigation controls across the pages.

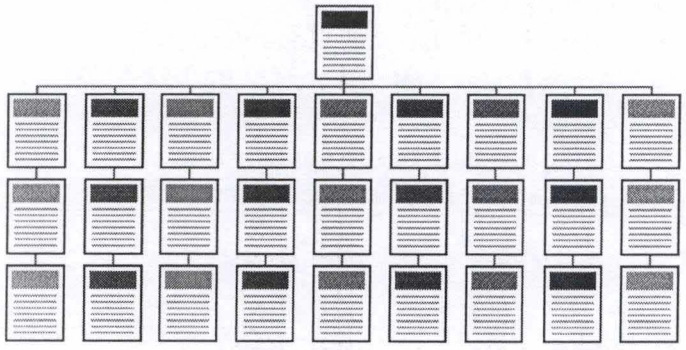


Figure 4-6. *A same page layout and graphic identity is reproduced across all the pages of the site*

2.4.2. Sobriety

Users are not impressed with complexity that seems gratuitous, especially those who may be depending on the site for timely and accurate work related information [Lynch99, p.16]. Pages crowded with text, links, graphics, and other components make it harder for users to find information on those pages. Many designers forget that white space is as important a component of a page as anything else.

Paradoxically, people complain about graphic design on the web being both dully and excessive. Users neither like to scroll through endless pages of text, without a break for the eye, all against the backdrop of a dismal grey background nor about pages displaying high-octane graphics with loudly crashing colours. It is essential for a designer to find the right balance according to the situation he is facing.

Technology allows designers to do so many neat things. Therefore it's often hard for them to resist showing all the wonders they can do with web technologies. From trite counters to moderately annoying, revolving animated GIFs to frustrating frames to the Java applets that, after taking eons to download don't add any functionality. Like graphics and other aspects of web site design, technologies should directly aid users in getting what they want out of a site [Rosenfeld98, p. 5].

2.4.3. Navigation

In designing complex web sites, it is particularly important to provide *context* within the greater whole [Rosenfeld98, p. 50]. Many contextual clues in the physical world do not exist on the web. There are no natural landmarks and no north and south. Unlike physical travel, hypertextual navigation allows users to be transported right into the middle of a large unfamiliar web site. Links from remote web pages and search engine result pages allow users to completely bypass the front door or main page of the web site. To further complicate matters, people often print web pages to read later or to pass along to a colleague, resulting in even more loss of context.

2.4.3.1. Building context

A few rules should always be followed to ensure that a site provides *contextual* clues [Rosenfeld98, p. 51].

First, all pages should include the site's name. This might be done as part of the title or header of the page. As a user moves through the levels of a site, it should be clear that they are still within that site. Carrying the graphic identity throughout the site supports such context and consistency. In addition, if a user bypasses the front door and directly accesses a subsidiary page of the site, it should be clear which site he or she is on.

Secondly, the navigation system should present the structure of the information hierarchy in a clear and consistent manner and indicate the location within that hierarchy.

In the context of a web course, those guidelines could apply as shown in figure 4-7.

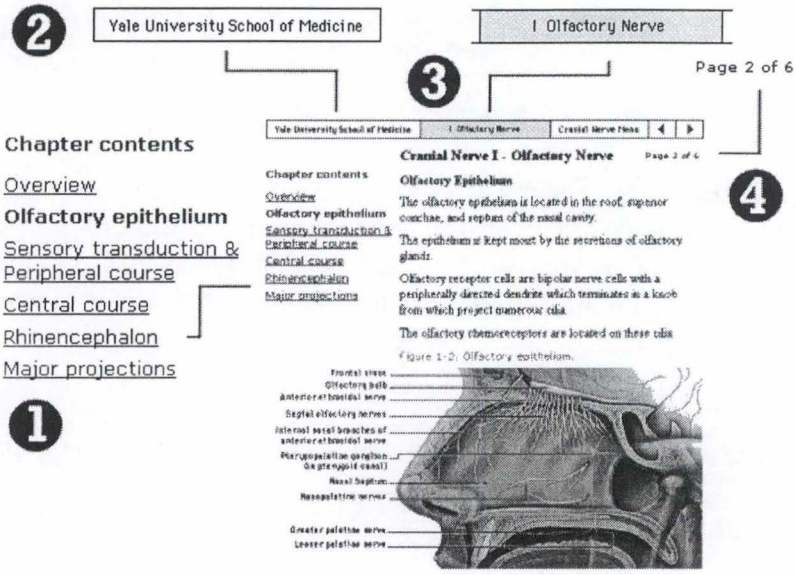


Figure 4-7. *On the side column (1), the position inside the chapter is clearly shown. The identity of the site is on the left of the navigation bar (2). At the centre of the navigation bar (3), the name and number of the chapter are indicated and highlighted. On the right, below the navigation bar (4), the page number as well as the total of pages in this chapter are shown.*

2.4.3.2. Hierarchical navigation

The simplest hierachical navigation system might consist of a graphical navigation bar at the top and/or bottom of each page on the site [Rosenfeld98, p.54]. In order to show the context (figure 4-8), the navigation bar should clearly highlight to which part of the hierarchy the current page belongs to.

Figure 4-8. *On Sun's web site, the position inside the whole site is clearly highlighted.*

More complicated navigation systems are composed of a variety of elements such as tables of contents, site maps and search engines which provide remote access to content within the organisation structure.

2.4.3.3. Sequential navigation

Inside a part of the site, a navigation system must be provided to allow the user to navigate sequentially [Lynch99, p.22]. This feature will often be implemented using "Previous" and "Next" buttons (figure 4-9).

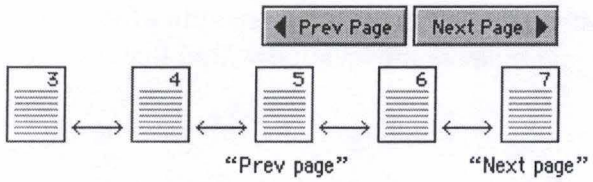


***Figure 4-9**. Sequential navigation system using "previous" and "next" buttons*

2.4.3.4. Conceptual navigation

Learning systems should have some amount of disorientation in order to facilitate exploration and learning [Mayes90]. While most designers are concentrating on predetermined hierarchical or sequential navigation, very little work has been done on providing a *conceptual navigation*. The problem is that simply following links to nodes does not necessarily provide effective learning. As a matter of fact, disorientation is required sometimes in order to explore and learn, simply by discovery.

In order to provide a conceptual navigation, the following features may apply:

- **Embedded links**

To provide exploration possibilities, a hypermedia course should provide links that are outside any hierarchical structure. The easiest way to create "learning by exploration" solution is to create *embedded links* [Rosenfeld98, p.56]. Every time that the user encounters a concept in a page that is defined in another page, a link to its definition should be provided. If the definition of the concept uses other concepts, they will also lead the user to their definitions and so on.

This navigation could be clearer if every embedded links contained in a page is reproduced in a dedicated space within that page.

***Figure 4-10**. An embedded link is reproduced in the left side column.*

- **Associative learning**

Navigation systems can be designed to support *associative learning* by featuring resources that are related to the content currently being displayed. For example, a page that describes a concept may include "see also" links to related information. The constant challenge in navigation system design is to balance this flexibility of movement with the danger of overwhelming the user with too many options.

In the context of a course, this associative learning feature could be implemented to provide links to concepts related to the one currently being displayed. The possibilities of such a feature using the Zephir product[2] have been tackled by Elise Remy and Michela Catizzone [Remy-Catizzone99],

2.4.4. Page layout

Graphic design creates visual logic and seeks an optimal balance between visual sensation and graphic information [Lynch99, p.53]. Without the visual impact of shape, colour, and contrast, pages are graphically boring and will not motivate the viewer. Dense text documents without contrast and visual relief are also harder to read, particularly on the relatively low-resolution screens of personal computers. The primary task of graphic design is to create a strong, consistent visual hierarchy in which important elements are emphasised and content is organised logically and predictably.

2.4.4.1. Visual contrast

The overall graphic balance and organisation of the page is crucial to drawing the reader into the content [Lynch99, p.54]. A dull page of solid text will repel the eye as a mass of undifferentiated grey, without obvious clues to the structure of the information. A page dominated by poorly designed or overly bold graphics or typography also will distract or repel users looking for substantive content. Designers should strike an appropriate balance between attracting the eye with visual contrast and providing a sense of organisation.

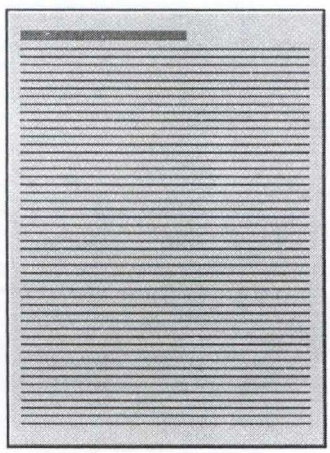


***Figure 4-11**. Dull; no focal points, no graphic structure*

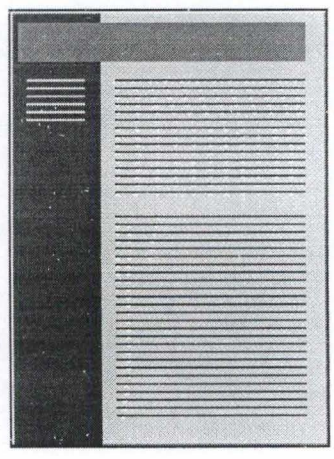


***Figure 4-12**. Stronger visual structure, better contrast*

[2] For more information, refer to the ARIADNE project web site at *http://ariadne.unil.ch/*

2.4.4.2. Layout grid

Current implementations of HTML do not allow the easy flexibility or control that graphic designers routinely expect from page layout software or multimedia authoring tools [Lynch99, p.62]. Yet HTML can be used to create complex and highly functional information systems if it is used thoughtfully. When used inappropriately or inconsistently, the typographic controls and inlined graphics of web pages can create a confusing visual jumble, without apparent hierarchy of importance (figure 4-13). Randomly mixed graphics and text decrease usability and legibility, just as they do in paper pages. A balanced and consistently implemented design scheme will increase readers' confidence in a site (figure 4-14).

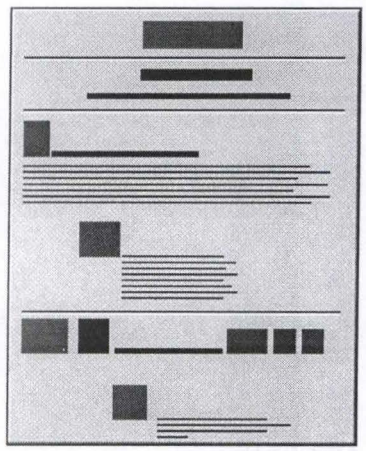


***Figure 4-13**. Poor page layout, no visual hierarchy.*

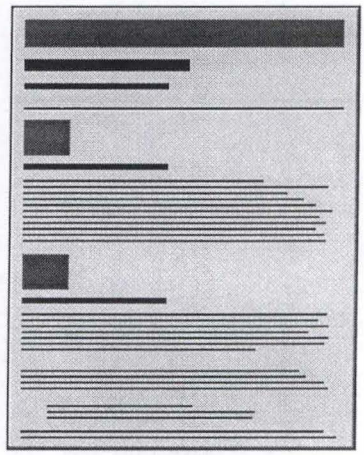


***Figure 4-14**. Better layout, balanced*

No one design grid system is appropriate for all web pages. A consistent, logical screen layout must be established, one that allows designers to "plug in" text and graphics without having to stop and rethink the basic design approach on each new page. Without underlying design grid, the project's page layout will be driven by the problems of the moment, and the overall design of the web site will seem patchy and confusing.

2.4.4.3. Additional guidelines

When designing web pages, it is also advised to consider the following guidelines:

- Reading becomes uncomfortable when there are too many words per line. If there is a long distance between the end of a line and the beginning of the next line, the eye has to make a significant shift to return to the left margin. Also, if the eye must traverse great distances on a page, the reader is easily lost and must hunt for the beginning of the next line. Quantitative studies show that moderate line length significantly increases the legibility of the text. Designers should use tables to limit the line length, ideally to ten to twelve words per line [Lynch99, p.68].

- A "scan column" along the left side of the page can be very useful. It does two jobs: it provides space for local links to related material (2.4.3.4), and gives visual relief by narrowing the right text column.

- Designers should avoid using frames [Nielsen96a; Lynch99, p.74]. Actually they present major drawbacks:
    - URLs stop working: the addressing information shown at the top of the browser no longer constitutes a complete specification of the information shown in the window.
    - If users create a bookmark in their browser they may not get the same view back when they follow the bookmark at a later date since the bookmark doesn't include a representation of the state of the frames on the page.
    - Many browsers cannot print framed pages appropriately.
    - Search engines have trouble with frames since they don't know what composites of frames to include as navigation units in their index.
    - Many web sites that offer users a choice between regular and framed versions have found that most users prefer frame-free designs.

- Web pages layout should be optimized according to the lowest screen resolution used by its potential audience [Lynch99, p.57].

3. Pedagogical interactive applications design guidelines

In order to develop pedagogical interactive applications, we decided to analyse existing applications so that we could extract some general principles. The reference we used was Microsoft Encarta 1999. We deduced from this analysis six principles. In addition to them, we will adapt the fundamental principle of *bootstrapping* to this context.

3.1. Direct manipulation

Direct manipulation offers the user the opportunity to interact with concepts, to manipulate them directly. This allows him to be an active processor of information while "entertaining" himself with the manipulation.

The figure 4-15 shows parts of a direct manipulation application explaining the concept of lever. The user can extend the lever until the weight of the little girl is able to balance the weight of the elephant.

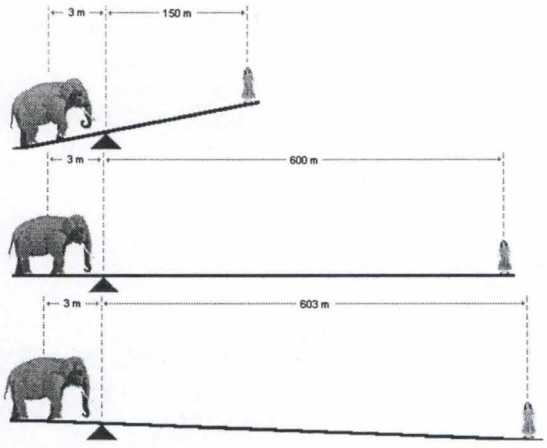


***Figure 4-15**. The concept of lever (Microsoft Encarta 99).*

3.2. Indirect manipulation

In an *indirect manipulation* application, the user doesn't interact with the concept itself but rather with its attributes. Manipulating an attribute has an influence on the concept representation.

Indirect manipulation helps the user to identify the attributes of a particular concept. Moreover, it gives him an understanding of the links existing between the attributes and the concept.

The figure 4-16 shows how the representation of a fractal tree depends on its orientation, its complexity or on the colour of its brunch and trunk. The user doesn't manipulate the tree but its attributes.

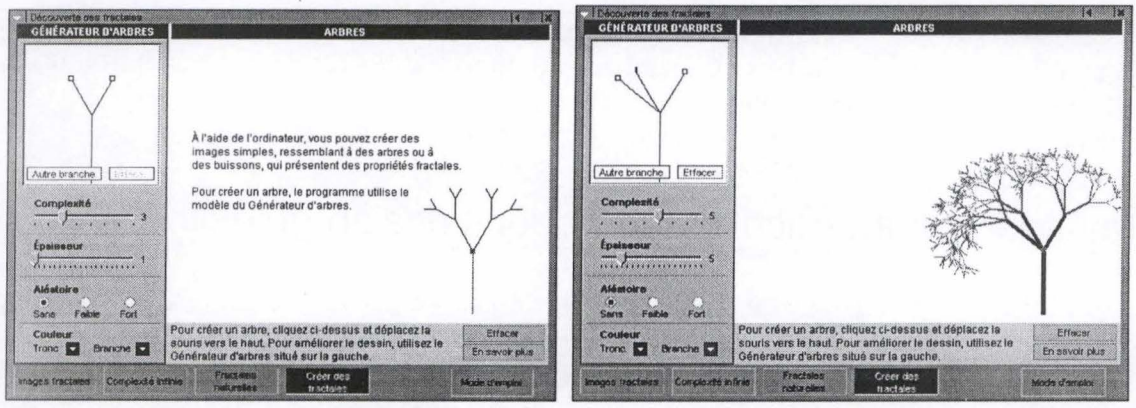


Figure 4-16. Two stages of the conception of a fractal tree (Microsoft Encarta 99).

3.3. Double reading

In a direct or indirect concept manipulation, the user doesn't always see clearly the consequences of his actions. Therefore it is important to give him a second *reading* (figure 4-17).

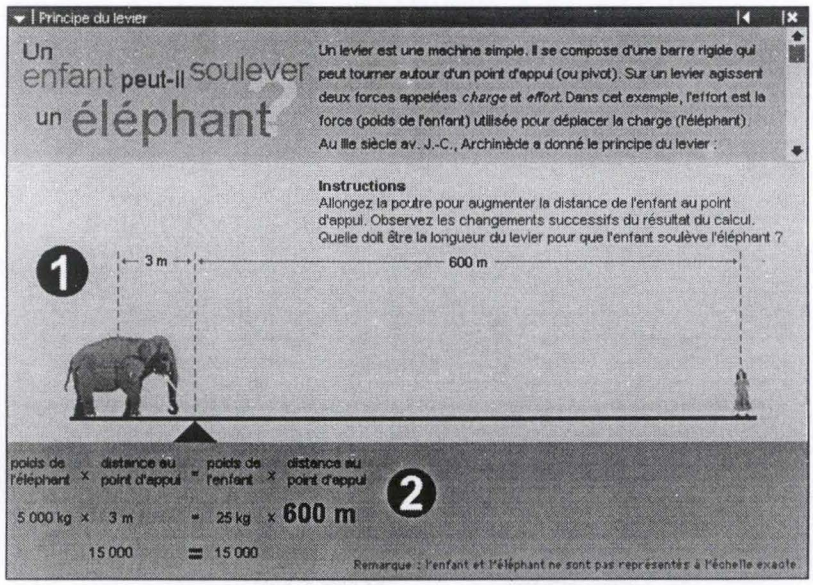


Figure 4-17. *When the user modifies directly the size of the lever (1), a mathematical interpretation of his action is displayed (2). (Microsoft Encarta 99).*

3.4. Related links

Related links are a collection of links to concepts that are in some way related to the concept currently displayed[3]. They offer the opportunity to go deeper in the understanding of a particular concept. The figure 4-18 represents an application designed to explain the concept of DNA chain. It offers links to related concepts such as "Genetic", "Genetic code" or "Nucleic and Deoxyribonucleic Acid".

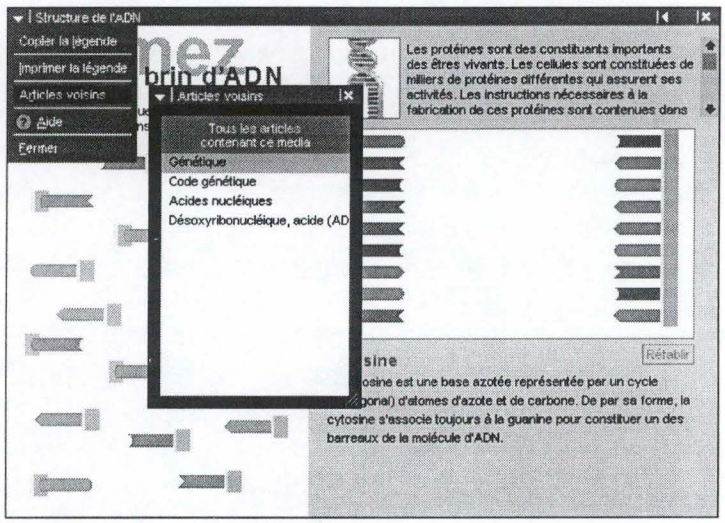


Figure 4-18. *The concept of DNA chain (Microsoft Encarta 99).*

3.5. Concepts manipulated definition

When presenting a concept, it is interesting to define the other concepts that are used in its manipulation.

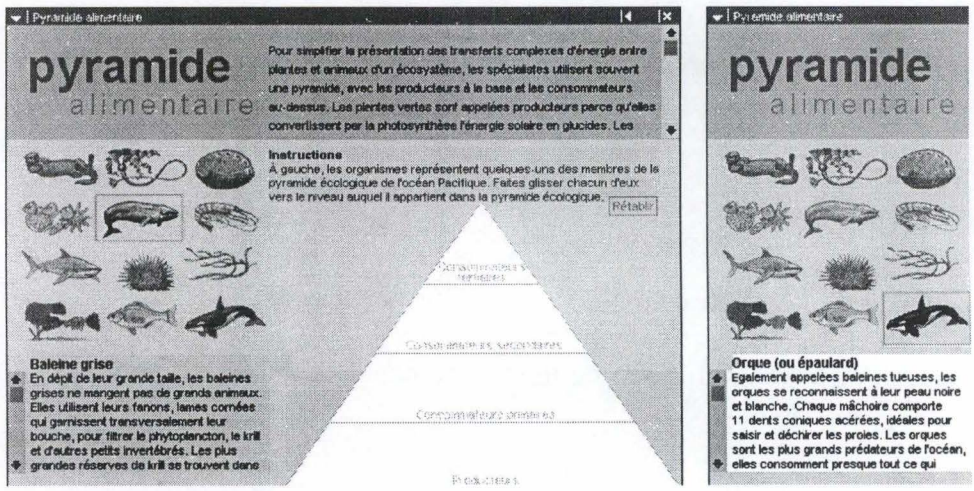


Figure 4-19. *The definitions of whale and orka in the concept of alimentary pyramid (Microsoft Encarta 99).*

[3] This principle is analogous to associative learning (2.4.3.4)

In the figure 4-19, aquatical animals are used to explain the concept of alimentary chain. When clicking on a particular animal representation, its definition is shown.

3.6. Instructions

In a concept manipulation application, it is also interesting to present the user some hints on the possible course of actions. It can show what is the best manipulation scheme to apply in order to get a better understanding of a concept. In the figure 4-20, an application presenting the fundamentals of probability advice the user to select the amount of tries and to press the "Lancer" button. It then recommends observing the occurrences of the number 7 over several launches.

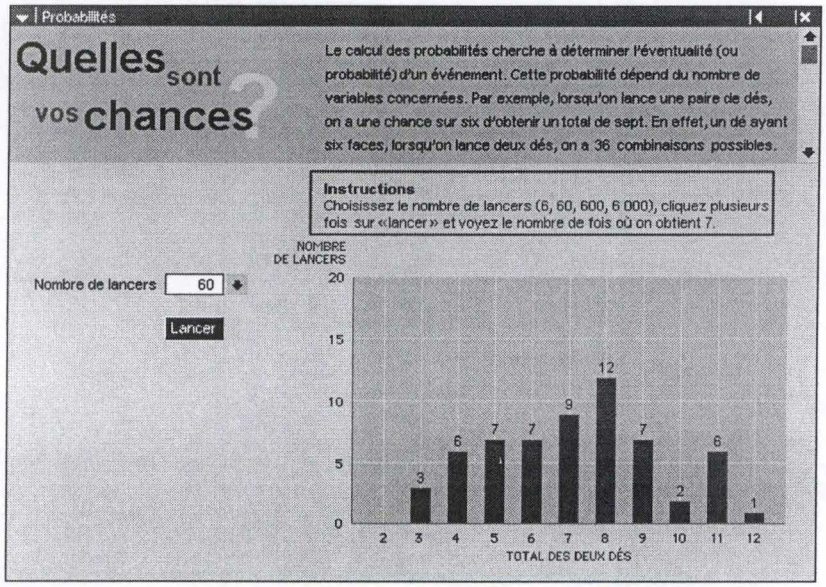


Figure 4-20. The concept of probability (Microsoft Encarta 99).

3.7. Bootstrapping

This fundamental principle states that every teaching process must follow the principles that it presents. For instance, a driving teacher can not tell his student to drive under 50 km/h in town while driving at the speed of 100 km/h !

4. Conclusion

In this chapter, we have researched some guidelines that will help us to develop the course notes as well as the interactive applications. As we have a solid theoretical basis at our disposal, we can now begin to describe the functionnalities of the environment. This is the purpose of the next chapter.

5

Analysis

1. Introduction

As described in Chapter 2, our part of the VESALE project consists on the development of the multimedia technologies database as well as the course notes related. In this chapter, we will describe the functionnalities of the environment that we will implement.

First, we will describe which interactive applications could be implemented. Then we will realise the conceptual analysis concerning the database. Afterwards, we will organise the course notes and will incorporate in this structure both the database consultation and the interactive applications. Finally, we will research the best way to present the course to the students.

2. Interactive applications

As stated in Chapter 3[1], it could be pedagogically interesting to provide interactive application to manipulate some concepts related to the Interaction Objects (IOs).

As IOs are by nature *interactive*, it seems obvious that allowing students to interact with them objects could considerably improve their learning. Therefore, we decided to provide an IOs manipulation application.

[1] See chapter 3 section 3.3.

As the IOs selection mechanism is quite complex and contains lots of rules, it seems also suitable to provide the students with an application that will allow them to acquire an intuitive feeling of the right IO to select in a particular case. Therefore we decided to develop a selection mechanism manipulation application.

2.1. IOs manipulation application

The first interactive application that we will develop is the IOs manipulation application.

2.1.1. Objectives of this application

Our objective in developing this application is not to teach how to *program* a User Interface with IOs, but is to teach what is their *potential.*

Consequently, the primary goal of this application is to give the user a good comprehension of the capabilities of some IOs. We want the user to be able to understand what can be done with those objects, and therefore, what they are good for. This application will be particularly important for complex objects.

2.1.2. Abstract and Concrete interaction objects

An Abstract Interaction Object (AIO) is an object used for the input and the display of data that the user can see, feel and manipulate. When working with AIOs, the focus is set on the behaviour of the object instead of on his graphic representation.

A Concrete Interaction Object (CIO) is an instance of an AIO. It is synonymous to a control, a physical interactor, a widget or a presentation object. Unlike AIOs, CIOs have a graphical representation depending on the graphical and presentation tool they belong to.

2.1.3. Principles followed

In the development of this application, we will follow those principles[2]:

- **Direct manipulation**

The best way to understand exactly how an AIO works is certainly to manipulate it *directly.* Therefore the application will provide a way of manipulating directly CIOs that are instances of AIOs.

- **Indirect manipulation**

Giving the student the possibility to manipulate directly some AIOs is certainly not enough if we want him to understand the full potential of those objects. An AIO presents some features that can't be manipulated directly such as its colour, its size or its orientation. Therefore, this application will provide a way of manipulating the *attributes* of the AIOs, and will immediately reflect those changes.

[2] See chapter 4 section 3 for their definitions

– **Double reading**

This *double reading* principle will be implemented as a consequence of the direct and indirect manipulation features.

If the student changes the value of an attribute, he will perceive the result of his action of two different ways. First, the value of the attribute will change and secondly, the object manipulated will react to this change. Identically, if the student manipulates directly an object, he will perceive the result of his manipulation on this object, and will secondly notice that an attribute has changed.

– **Related links**

A few *links* to related concepts should be provided by the application. For instance, links to the manipulated AIO description, to ergonomic rules or to the selection rules related to some AIOs could appear on the interface application.

– **Concepts manipulated definition**

Some concepts manipulated will be *defined* whether on a linked web page or as "tooltips" (or hints). The main manipulated concepts in this application will be the attributes.

– **Bootstrapping**

At the level of this application, the *bootstrapping* principle will be implemented by giving the user the opportunity to manipulate the CIOs that have been used in order to build the application interface.

2.2. AIOs selection trees manipulation application

The second interactive application that we will develop is the selection tree manipulation application.

2.2.1. AIOs selection tree

The concept of selection tree that we will use is excerpt from Jean Vanderdonckt's Ph. D. thesis [Vanderdonckt97, p. 151-206]. It provides a mechanism for the selection of IOs [Bodart-Magnier99][3].

[3] The section 4.1.1. of appendix A provides a more detailed definition of the concept.

The Figure 5-1 represents a selection tree used for the selection of AIOs.

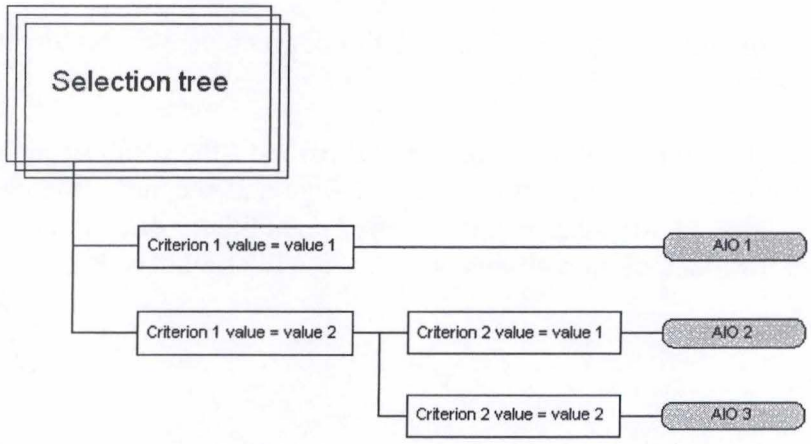


Figure 5-1. *A selection tree*

A selection tree consists of *nodes* and *leaves.* A definition of those concepts follows.

– **The nodes**

A *node* is a decision point where the tree is divided into several sub-trees. (figure 5-2). When reaching a node, the user has to decide what is the value of the criterion associated with the node, and then go through the sub-tree connected to this particular value.

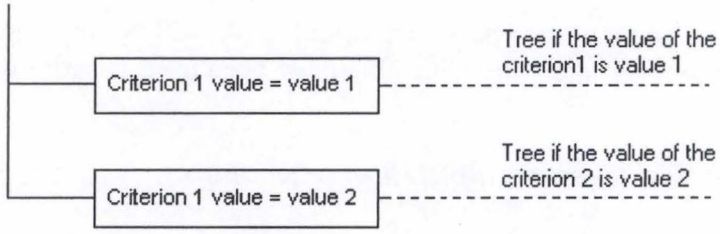


Figure 5-2. *A node*

For instance, if the value of the criterion 1 is equal to value 2 (figure 5-3), then the upper part of the tree is irrelevant and the user can go trough the lower part of the tree.

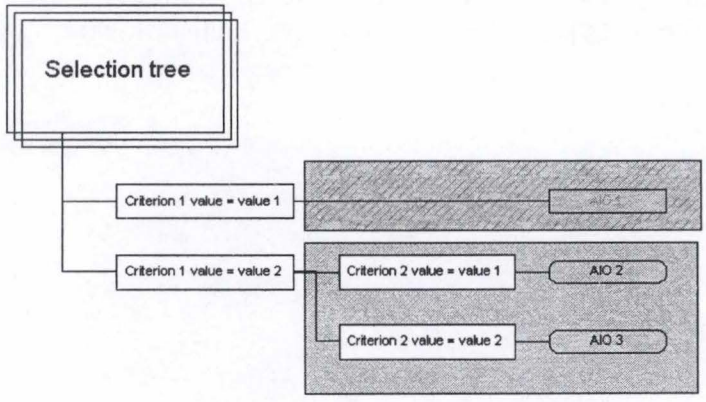


Figure 5-3. *A selection tree*

- **The leaves.**

A *leaf* represents an AIO (figure 5-4). When the user reaches a leaf, it means that the selected AIO for his particular needs is the AIO represented by the leaf.

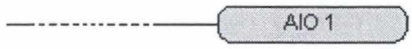


Figure 5-4. *A leaf*

2.2.2. Split selection tree

Even if we adapted Vanderdonckt's selection tree [Vanderdonckt97, Appendix D p.60-68][4] during our internship at UPE in order to include some more suitable AIOs[5], we decided to take his trees without doing any modifications to develop this application. Anyway, while developing it, we will consider the fact that the selection tree used might be adapted to include new objects. A mechanism will be provided to allow the teacher to change easily the tree handled by this application.

The size of the entire tree being quite considerable, we decided to split it into eight different selection trees (named D1, D2, ..., D8) according to the type of the values handled. The Figure 5-5 shows the *top-level* selection tree.

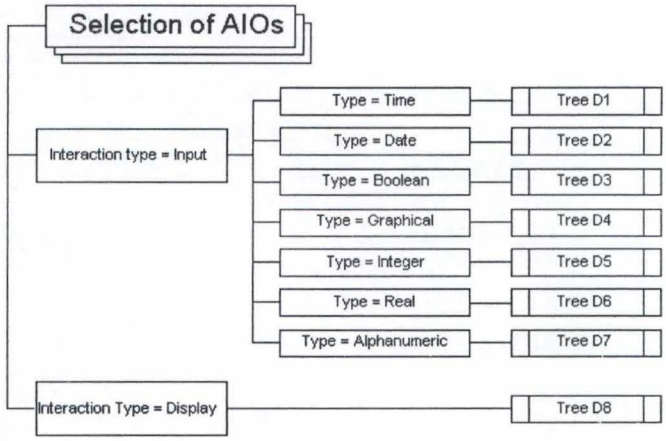


Figure 5-5*. The selection tree and its sub-trees*

Consequently, the student won't navigate inside the entire tree, but inside one of the height sub-trees defined.

2.2.3. Objectives of this application

The main objective that we want to achieve is that a student who has used this application for some time will know exactly which AIOs he must or mustn't use when developing an interface. He would not be able to do so because he has learned the trees by heart but because he got accustomed with each selection criteria. By acquiring *reflexes*, he will then be able to select the most suitable AIO for each particular case.

---

[4] See appendix A section 8.1 for a graphical version of Vanderdonckt's trees.
[5] See appendix A section 8.2 for a graphical version of our trees.

2.2.4. Principles followed

In the development of this application, we will follow those principles[6]:

- **Indirect manipulation**

An *indirect manipulation* feature will be provided. The key concepts that the user will manipulate are the selection criteria[7]. The student will be able to assign values to some selection criteria and will immediately see the result of this indirect manipulation. This result will be displayed as a partition of the AIOs included in the selection tree. The AIOs will be divided in two distinct sets; one for the AIOs that can be selected according to the values assigned to the selection criteria and one for the AIOs that can't be selected.

For instance, if the user assigns the value "Low" to the "Density" criterion in the tree displayed in figure 5-6, then the AIOs set can be divided into two sub-sets:

- one containing the AIOs that can be selected, namely "AIO 1", "AIO 3", "AIO 5" and "AIO 7"
- one containing the AIOs that can't be selected, namely "AIO 2", "AIO 4", "AIO 6" and "AIO 8"

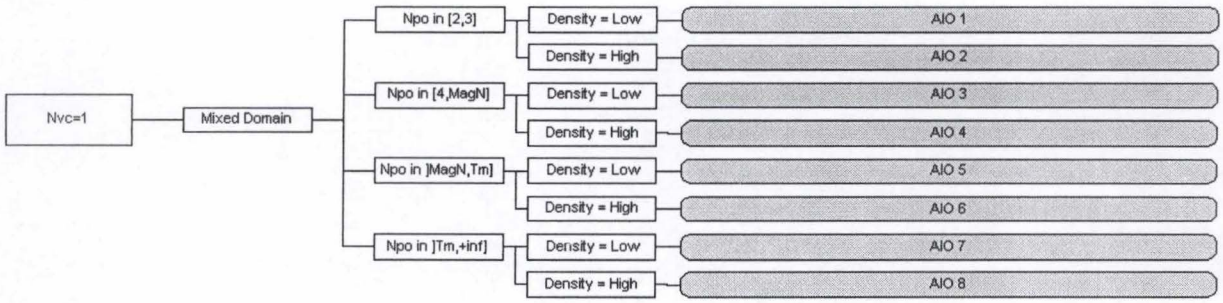


Figure 5-6. *A sub-tree of the main selection tree*

- **Double reading**

A *double reading* feature will also be provided. As soon as a student assigns a value to a specific criterion, the application will indicate the implications in term of expected features for the AIOs that can be selected.

For instance, if the student assigns the value "Unknown" to the "Domain" criterion, the application will display the following message: "The AIOs that can be selected provide the possibility for the user to add a value."

The combination of this *double reading* principle with the *indirect manipulation* one will provide the following feature: the student will know that the objects displayed in the set of AIOs that can be selected have the features described by the double reading messages. For instance, in the example displayed in the figure 5-6, the assignment of the "Low" value to the

[6] See chapter 4 section 3 for their definitions
[7] The criteria used in the selection trees are fully explained in [Bodart-Magnier99] section 4.1.2.1.

"Density" criterion will display the following double reading message: "The AIOs that can be selected can take up a lot of room space". Therefore the user will know that the objects displayed in the set of AIOs that can be selected take up a lot of room space.

– **Bootstrapping**

The principle of *bootstrapping* will also be implemented for the selection of the IOs used in this application interface. This means that the components used in the interface will have to be selected according to the selection tree handled by the application.

Consequently, the interface of this application might be used as an illustration in the course notes related to the concept of selection tree.

– **Concepts manipulated definition**

The concepts manipulated will be *defined* inside the application interface. The main manipulated concepts are:

- The selection criteria. As those concepts are fully explained in the course notes, a link to a page illustrating each criterion should be provided.
- The AIOs. The AIOs have often names that are not very expressive like "Scrollable Drop-Down Graphic List Box" or "Check Box + Editable Non-Contextual Accumulator + Group Box". Those concepts should therefore be illustrated by a picture representing an instance of those AIOs, as well as a link to their definition pages.

– **Related links**

A few links to related concepts should be provided by the application. For instance, links to the manipulation application, to ergonomic rules or to the selection rules related to some AIOs could appear on the interface application.

– **Instructions**

Some instructions concerning the manipulations that the user can do will be displayed. For instance, the application could tell the user that if a particular AIO is displayed in the list of the objects that can't be selected, the assignment of a specific value to a criterion will make that this object could then be selected.

Moreover, if an AIO is displayed in the list of the AIOs that can be selected, the application could tell the user that in order to make this object the only one that can be selected, some criteria assignments should be made.

3. Interaction Objects database

The database we have to develop will contain information about interaction objects. The purpose of gathering all this information about IOs in a database instead of creating static web pages is to allow the teacher to easily insert the description of new objects.

First, we will decide on which information should be stored in the database. Secondly we will analyse which extensions could be brought in the future in order to link this database to the other bases of the VESALE project

3.1. Conceptual analysis

In this section we will describe all the entities and associations related to the concepts of AIOs, CIOs, attributes, events, primitives and ergonomic rules[8].

3.1.1. The AIO entity type

An *AIO* entity represents an Abstract Interaction Object and contains the following attributes:

- Idaio: a short name which identifies the AIO
- Enname: the English name of the AIO
- Frname: the French name of the AIO
- Description: the description of the AIO
- Category: the category the AIO belongs to. The possible values are "action", "scrolling", "static", "dialog", "control" and "feed-back"
- Manipurl: the URL of the manipulation application (if available)

The figure 5-7 gives the graphical representation of the AIO entity type.

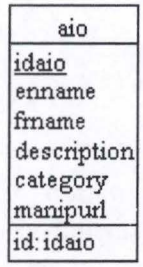


Figure 5-7. *The AIO entity type*

[8] The entire conceptual schema is in appendix B section 1.

The table 5-1 gives an example of a possible AIO entity.

Idaio:	LIB
Enname:	List Box
Frname:	Liste de sélection
Description:	A List Box displays the possible choices in which the user can select one or more than one.
Category:	Control
Manipurl:	/oia/oi/apps/listbox.class

Table 5-1. *An instance of the AIO entity type*

3.1.2. The Attribute, Event and Primitive entity types

An *Attribute* entity represents a management attribute of an AIO. It contains two attributes: a name (which identifies it) and a description. The table 5-2 gives an example of a possible instance of the Attribute entity type.

Name:	ATTR_NB_ITEM_DISPLAYD _LIST
Description :	number of items displayed in a list

Table 5-2. *An instance of the Attribute entity type.*

An *Event* entity describes an action that the user can execute on an AIO. It contains two attributes: a name (which identifies it) and a description. The table 5-3 gives an example of a possible instance of the Event entity type.

Name:	EVT_ITEM_SELECTED_LIST
Description :	Event generated when the user selects an item in a list

Table 5-3. *An instance of the Event entity type.*

A *Primitive* entity describes an external manipulation primitive used to interact with an AIO. It contains two attributes: a name (which identifies it) and a description. The table 5-4 gives an example of a possible instance of the Primitive entity type.

Name:	PR_ADD_ITEM_LIST
Description :	Adds an item to the list

Table 5-4. *An instance of the Primitive entity type.*

An AIO can have from 0 to N Attributes, Events or Primitives. On the other hand, an Attribute, Event or Primitive can belong to from 0 to N AIOs. The figure 5-8 illustrates the relationships between the AIO entity type and the Attributes, Events and Primitives entity types.

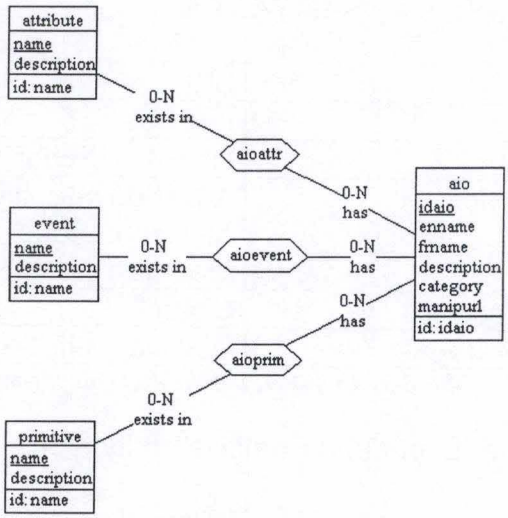


Figure 5-8. Relationships between the AIO entity type and the Primitives, Attributes and Events entity types.

3.1.3. The Inheritance association type

An AIO can *inherit* from 0 to N AIOs. This relationship is illustrated is the figure 5-9.

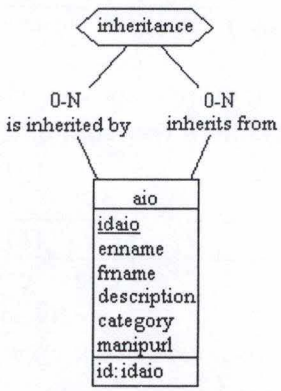


Figure 5-9. The inheritance association type

This inheritance association type between the AIO entity type and itself describes the fact that an AIO is an aggregate of other AIOs and, therefore, inherits from the characteristics of its aggregating objects. The figure 5-10 shows how the "Combo Box" AIO is an aggregation of two AIOs.

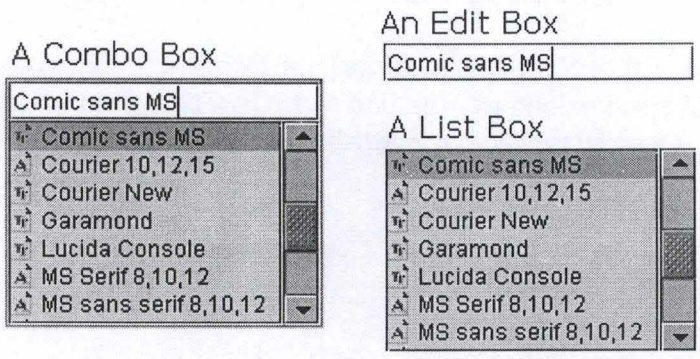


Figure 5-10: A Combo Box and its aggregating AIOs

3.1.4. The CIO entity type

A *CIO* entity represents a concrete interaction object and contains the following attributes:

- Idcio: a number which identifies the CIO along with the instantiation association.
- Name: the name of the CIO
- Description: the description of the CIO
- Graphtool: the graphical tool of the CIO
- Prestool: the presentation tool of the CIO

The figure 5-11 gives a graphical representation of the CIO entity type as well as the instantiation association type.

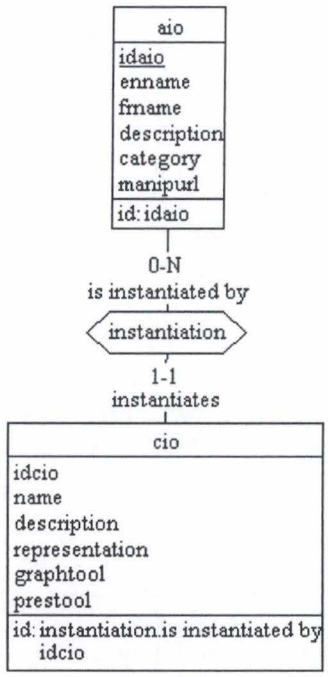


Figure 5-11*. The CIO entity type and the instantiation association type*

3.1.5. The Ergrule entity type

An *ergrule* entity represents an ergonomic rule. It contains the following attributes:

- Idrule : a number which identifies the rule
- Rule: the content of the rule
- Posex: a positive example illustrating a correct application of the rule
- Posexillustr: an illustration of the positive example
- Negex: a negative example showing the rule violated
- Negexillustr: an illustration of the negative example
- Justification: the justification of the rule

An ergonomic rule can concern from 0 to N AIOs. The ergonomic rule concept is much more complex that what appears here. As part of this work, we limited ourselves to a basic ergonomic rule entity type. This entity type will be significantly enhanced as part of a next stage in the VESALE project.

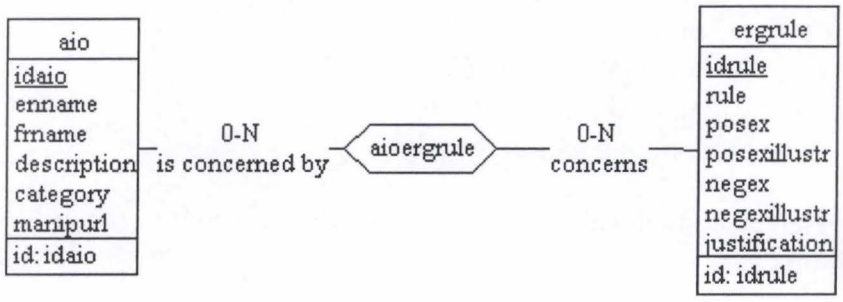


Figure 5-12. *The CIO entity type and the instantiation association type*

3.2. Extensions

One of most important features of the VESALE project is the interconnection of the databases. For instance, the Interaction Objects database could easily be linked with the video sequence database which is not yet implemented.

The reasoned cases database could also be easily linked to the IOs database. It could indeed be quite interesting to present to the user a few pictures showing a particular interaction object used (either positively or negatively) in a particular context. This reasoned cases database is being implemented at the moment by Rudy Michiels and Gaëtan Prévot [Michiels-Prévot99].

4. Information architecture

At this stage, we will organise the course notes and will incorporate in this structure both the database consultation and the two interactive applications.

First, we will divide the course into small pieces of information. Secondly, we will organise those *chunks* according to the pedagogical scenarios defined in chapter 3[9]. Finally, we will say a few words about how we have organised a conceptual navigation[10]

[9] See chapter 3 section 4
[10] See chapter 4 section 2.2.2.4.

4.1. Course chunking

We have divided the course into information chunks according to the concepts they present[11]. One chunk will be at a later stage displayed on one web page. Ideally, one chunk will contain only one concept. Anyway, in some cases, the concept description will be too long to fit on the one and a half screen limit stated in the guidelines. In that case, we will split the concept into sub-concepts.

For instance, the concept of AIO was too long to fit on one page. The paper course first defined the concept of AIO itself. It then explained how to describe an AIO. Finally it presented the categories of AIOs. We have decided that the web version of the AIO concept definition will be subdivided into three information chunks. A first chunk is the concept itself, which is then subdivided into two sub-concepts: the description and the categories (figure 5-13).

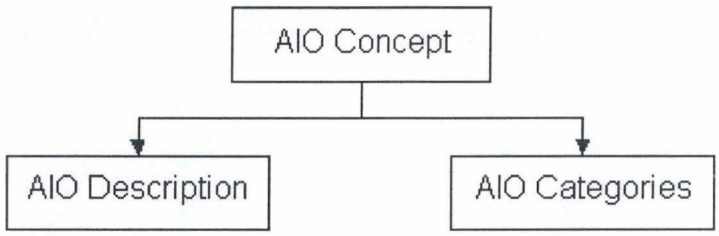


Figure 5-13. *The AIO concept information chunk and its two sub-concept.*

We have applied the same process to the course concerning the AIO selection. Finally, we obtained about forty chunks of information.

4.2. Pedagogical scenarios

Now that we have divided the course into information chunks, we have to organise them into architectures. As stated in chapter 3[12], we will organise the chunks according to two distinct scenarios. We will also introduce a third scenario that corresponds to a particular VESALE application: the apprenticeship as part of the live teaching.

4.2.1. Scenario 1: Objectivism oriented

In this scenario, we will approximately follow the paper course structure. The two interactive applications will take place in two distinct sections and will follow their related theory.

This scenario is composed of a hierarchical architecture[13] along with a sequential one[14]:

[11] See chapter 4 section 2.2.1.
[12] See chapter 3 section 4
[13] See chapter 4 section 2.2.2.2
[14] See chapter 4 section 2.2.2.3

- **Hierarchical architecture**

The top level of the hierarchical architecture is shown in figure 5-14. You will find the complete hierarchical architecture of this scenario in Appendix C[15].

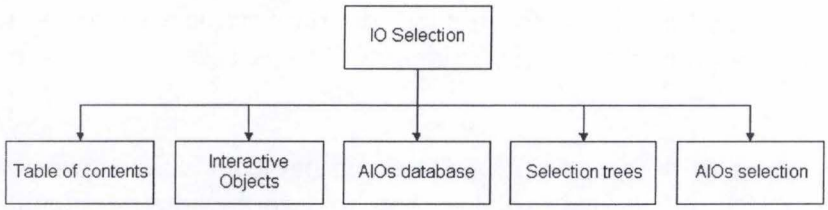


***Figure 5-14** The top-level hierarchical architecture of the Objectivist scenario*

- **Sequential architecture**

The top level of the sequential architecture is shown in figure 5-15. You can find the complete sequential architecture in Appendix C[16].

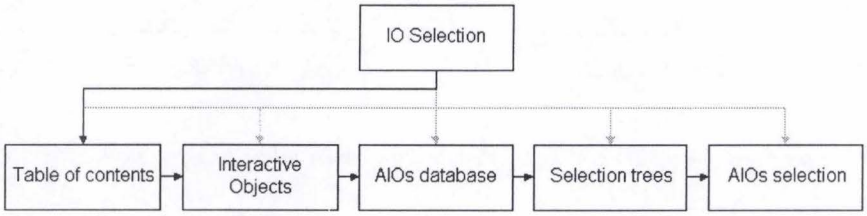


***Figure 5-15** The top-level sequential architecture of the Objectivist scenario*

4.2.2. Scenario 2: Constructivism oriented

This scenario will divide the course into two main parts: the *interactive objects* and the *AIOs selection*. The first part will begin with the IOs manipulation application while the second will begin with the selection tree application.

This scenario is composed of a hierarchical architecture along with a sequential one:

- **Hierarchical architecture**

The top level of the sequential architecture is shown in figure 5-16. You can find the complete sequential architecture in Appendix C[17].

[15] See appendix C section 1.1
[16] See appendix C section 1.2
[17] See appendix C section 2.1

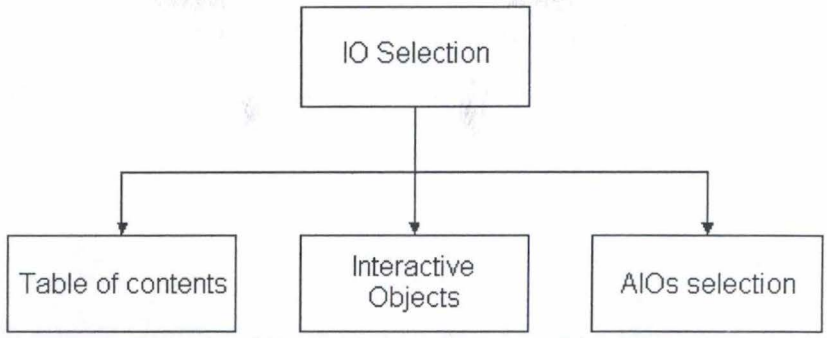


***Figure 5-16**. The top-level hierarchical architecture of the Constructivist scenario*

- **Sequential architecture**

The top level of the sequential architecture is shown in figure 5-17. You can find the complete sequential architecture in Appendix C[18].

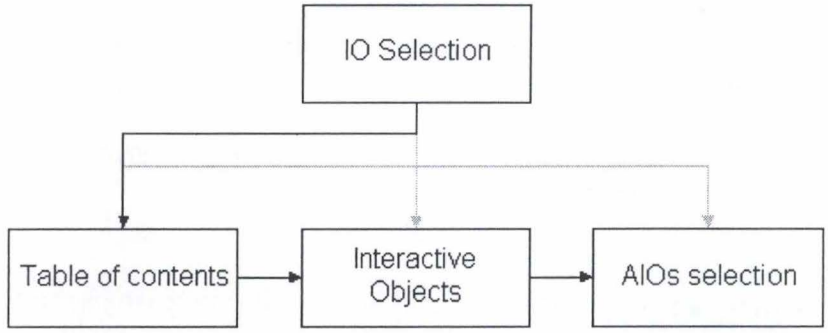


***Figure 5-17** The top-level sequential architecture of the Constructivist scenario*

4.2.3. Customisable life teaching scenarios

If a teacher uses the VESALE environment during a lecture, he could want to reorganise the course according to the currently taught matter.

One solution could be to provide the teacher with a dedicated interface in which he can select and order the pages or applications of the course that he wants to present during his next lecture. Once the pages are selected and ordered, the system could generate a specific scenario including a hierarchical and a sequential navigation containing only the content selected by the teacher. We don't intend to implement this features and, therefore, we won't go any further in its development.

4.3. Conceptual navigation

The conceptual navigation is independent from any navigation scheme and therefore is common to all scenarios. It is supported by embedded and associative learning links[19].

Every time the user encounters a concept that is defined in another page, an *embedded link* to its definition page will be provided.

---

[18] See appendix C section 2.2
[19] See chapter 4 section 2.4.3.4.

Associative learning intend to organise the course according to a conceptual graph. We have decided not to build this conceptual graph as part of this thesis, and consequently, not to include those conceptual links in the course.

5. Page structure

Now that we have content organised in scenarios, we have to think about a consistent way of presenting it to the learner. The web design guidelines described in chapter 4[20] will help us in developing a web page structure that will be *usable*, i.e. memorable, easy to learn and pleasing[21].

5.1. Site identity and sobriety

A student reading the course has to be fully focused on the content. He should not have to adapt himself to each page of the site. Each individual page must present a same layout grid, navigation scheme and graphical style[22].

Furthermore, it is important to think the site in term of *sobriety*[23]. Once again, we must not forget that our audience is composed of students who want an effective design that serves *usability*.

Site identity and sobriety are two capital elements to keep in mind when thinking about the navigation and the pages layout.

5.2. Navigation

Some elements are fundamentals in order to build context[24] and must therefore be included on each page of the site.

First, all pages must show a logo of the site providing a link to its home page. This logo could be either the university's logo or a specific logo for the project. This second solution seem to be better since a link to the VESALE home page on the university's logo can be confusing for users.

Secondly we will provide hierarchical, sequential and conceptual navigation systems inside the pages.

5.2.1. Hierarchical navigation system

The hierarchical navigation system of a chapter will present the structure of the information in a clear and consistent way and indicate the location within the hierarchical organisation.

[20] See chapter 4 section 2
[21] See chapter 4 section 2.1.
[22] See chapter 4 section 2.4.1
[23] See chapter 4 section 2.4.2
[24] See chapter 4 section 2.4.3.1

In order to ensure that the student can build a good mental model of a chapter structure, we will provide the following information on every pages:

- The chapter name presented as a link to the chapter's first page
- The current section name
- The position of the page in the chapter hierarchy presented as a hypertext menu
- The numeric position of the page in the chapter

In addition to the elements described above, the hierarchical navigation system will include a table of contents for the chapter.

5.2.2. Sequential navigation system

The sequential navigation system will allow the user to view the next as well as the previous page in the sequential organisation of the course.

5.2.3. Conceptual navigation system

A dedicated place in the page structure will be reserved in order to bring to the fore the embedded and associative learning links.

If a link redirects the user to an external web site, it must be clear for the user that following this link will take him to another site. This can be achieved by preceding external links with a special icon.

5.3. Page layout

Now that we have identified all the elements that will be displayed on each page, we have to find a way to organise them in a layout grid[25].

5.3.1. Visual contrast

As suggested in Chapter 4[26], we will try to build a layout with a strong visual structure and a good contrast. We will adopt a structure divided into four major parts: header, left column, text body and footer as shown in figure 5-18.

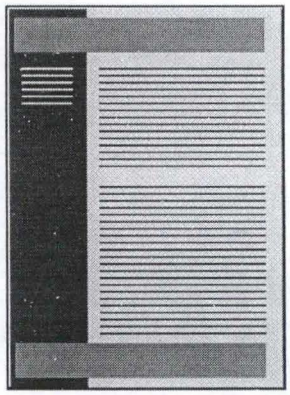


Figure 5-18. *The general visual aspect of a page*

[25] See chapter 4 section 2.4.4.
[26] See chapter 4 section 2.4.4.1.

5.3.2. Layout grid

Now that we know which elements will be part of the pages structure and that we have a general idea of the visual layout we can build a layout grid that will be used throughout the site[27].

The logo, being the guaranty of the site identity will be placed in the top left corner of the page. Therefore, the logo will be a sort of intersection between the left column and the header section.

The header section will regroup simple navigation information helping the user to know which page he is reading. Additionally, the header section will provide the user with the sequential way of browsing the course. Consequently, this section will regroup the chapter name, the page name, the numeric position of the page in the chapter and the sequential navigation bar.

The left column will regroup more complex navigation information. It will provide the user with the exact position of the page inside the chapter as well as all kinds of related content links. Consequently, this section will regroup the section structure and the embedded links.

Finally, the footer section will provide copyright information as well as a copy of the sequential navigation bar.

The figure 5-19 shows our page structure.

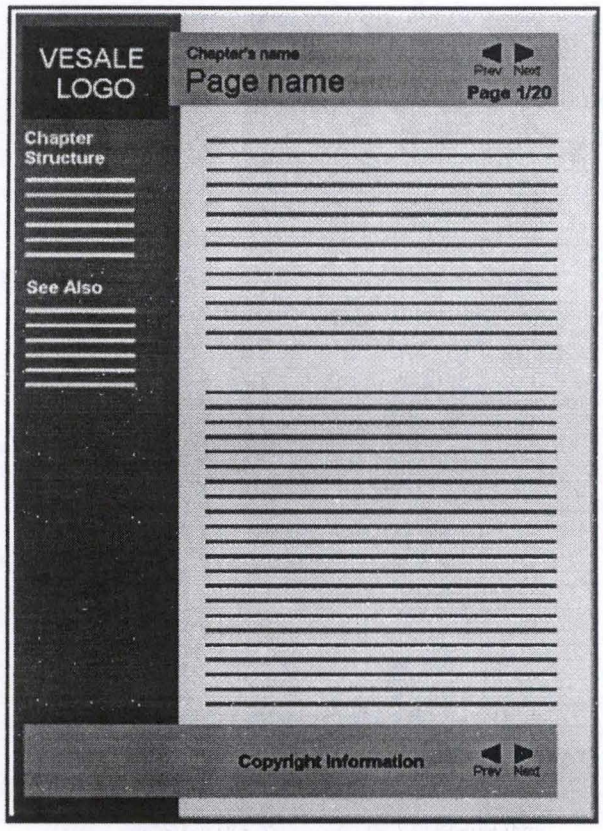


Figure 5-19. *The layout grid of the site pages*

[27] See chapter 4 section 2.4.4.2.

5.4. Dynamic navigation generation

As pleasing as it may seem, this navigation scheme suffers from its apparent complexity and its total lack of flexibility. For instance, if the teacher wants to add a page in the middle of a chapter, he has to change the surrounding pages sequential navigation bars and the section structure in almost every pages as well as the table of contents.

Keeping in mind that we have to improve flexibility without sacrifying usability, we must find a way to generate all the navigation elements dynamically.

The best way to achieve this goal is to generate all navigational parts of a page according to one unique table of content. Therefore, adding a page to the course will be as easy as modifying this unique table.

6. Conclusion

In this chapter, we described the elements that we will develop as part of this thesis. What we need now is suitable technologies in order to implement them. This is the purpose of the next chapter.

6

Technology choices

1. Introduction

In chapter 5, we have described the elements that we will develop as part of this thesis. In this chapter explains we will explain how we selected the most suitable technologies in order to implement those elements.

While we were working on this thesis, the technologies that will be used for the VESALE project were not yet selected. Therefore, we had to make our own technological choices. Nevertheless, we tried as far as possible to use *platform independent* technologies so that the result of this work could be effectively incorporated into the project.

First, we will explain why our architecture is typically a three-tier one. Then, we will select technologies for each tier, namely: client-side, middleware and database.

2. Three-tier architecture

The architecture of our project is typically a *three-tier architecture* as shown in figure 6-1.

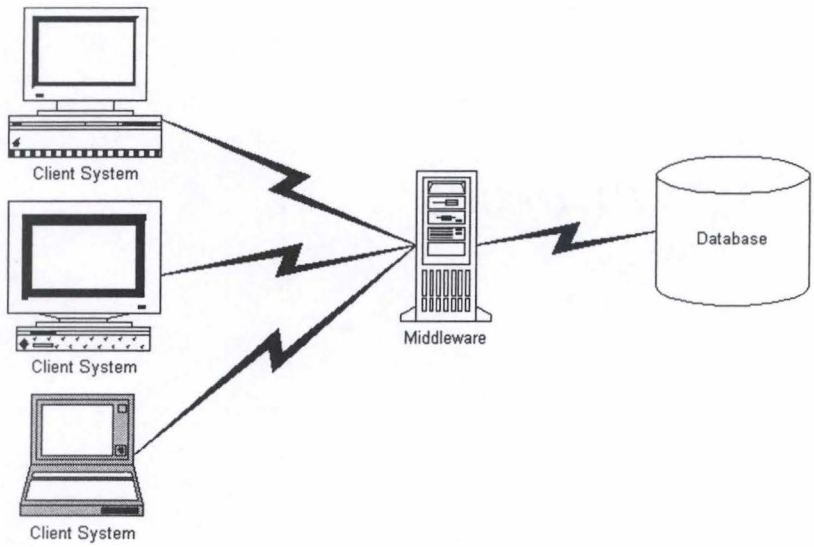


Figure 6-1. *A three-tier architecture*

The first tier is composed of all the client systems making HTTP[1] request to the web server placed in the middle-tier. As the HTTP protocol *is platform independent*, the client systems can be of any kinds as long as they support this protocol at the top of the TCP/IP protocol.

The web server composes the middle-tier along with the technology used to serve and generate web pages. The middleware is linked to both the client systems using the HTTP protocol and to the DBMS[2].

The third tier is composed of the database. Its goal is to store data and to respond to the queries executed by the middle-tier.

3. Client-side technologies

In order to reach our objectives, we need client-side technologies able to do the following things:

- To present information content to the user. This point has already been discussed and the technology used will be HTML.
- To provide a way to execute applications such as the IOs manipulation application and the selection trees application.

[1] HyperText Transfer Protocol
[2] DataBase Management System

As we said before, the client systems can be of any kinds as long as they support the HTTP protocol at the top of the TCP/IP protocol and provide a way of presenting HTML formatted information, namely a browser. As long as those conditions are respected, our system will be cross-platform.

Today's crop of web browsers are far more than data readers [Goodman98, p.15]. Each one includes a highly customised content rendering engine, a scripting language interpreter, a link to a custom Java virtual machine (JVM), security access mechanisms, and connections to related software modules. The instant we decide to author content that will be displayed in a web browser, we must concern ourselves with the capabilities built into each browser. Fortunately, there is a certain level of interoperability due to industry-wide standards.

The VESALE Project specifications states that the browser used will be Netscape Navigator 4.x. As far as possible, we will try to stay *cross-browser*. Nevertheless, it is possible that one of our applications run perfectly on Netscape Navigator and shows an undesired behaviour when executed on Internet Explorer. Concerning the course notes, we will try to create browser-independent pages.

3.1. Hypertext Mark-up Language

Hypertext Mark-up Language (HTML) is the *lingua franca* for publishing hypertext on the World Wide Web [Raggett99]. It is a non-proprietary format based upon W3C recommendations and uses tags to structure documents.

HTML can be created and processed by a wide range of tools, from simple plain text editors to sophisticated "What You See Is What You Get" (WYSIWYG) authoring tools. The most powerful tools are situated between those two extremes; they offer the advantages of a WYSIWYG editor as well as the flexibility of a plain text editor. We will use such a tool in order to build our pages, namely Dreamweaver 2.0 by Macromedia.

3.2. Cascading Style Sheets

As stated in our analysis of the problem[3], we want a site that possesses its own *identity*. This identity concerns as far the graphical aspect as the typographical one. To ensure that each page of the site is rendered equally, using the same typographical conventions, we have decided to use the Cascading Style Sheet (CSS) feature. This CSS technology is implemented in both 4.x versions of Internet Explorer and Netscape Navigator.

A *style sheet* is a definition of how content should be rendered on the page [Bos99]. The link between a style sheet and the content it influences is either the tag name of the HTML element that holds the content or an identifier associated with the element by way of an attribute (CLASS attribute). That's how the separation of style from content works: the content is ignorant of the style and the style is ignorant of the content.

[3] See chapter 5 section 5.1.

The following example (figure 6.2) shows how the <H1> tag is normally rendered in Netscape Navigator and how it looks like using a style sheet definition.

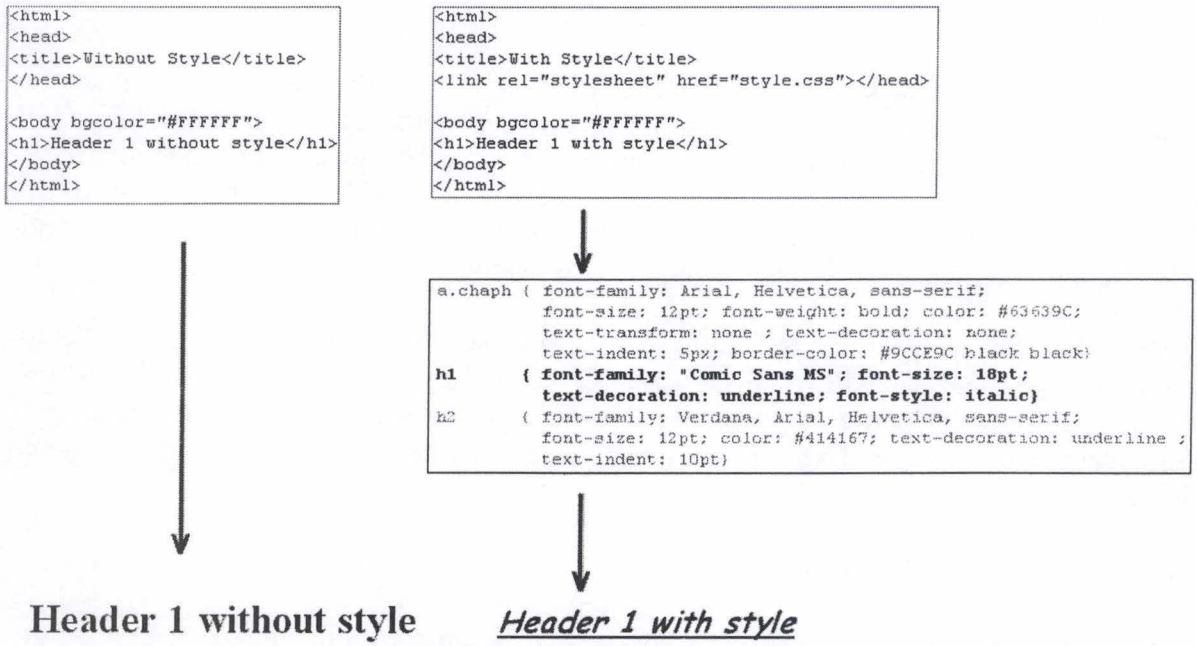


Figure 6-2. A HTML header with and without the style sheet definition

3.3. Client-side applications: Java

The best way to present an application to the user without breaking the continuity of the web course is to execute it directly in the browser.

The two major technologies to do such a thing are Microsoft's ActiveX and Sun's Java. As we explained earlier, the projects specifications states that Netscape Navigator will be favoured to the detriment of Internet Explorer. As ActiveX is a Microsoft proprietary standard and can only be used freely with Internet Explorer[4], we can not use this technology. Consequently, we will use Java to implement the interactive applications. Those applications will be implemented as Java applets using components from the Swing library.

3.3.1. Java

Java is two things: a platform[5] and a simple, architecture-neutral, object-oriented, portable, distributed, high-performance, interpreted, multithreaded, robust, dynamic and secure high-level programming language[6] [Sun99a].

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler, a Java program is translated into an intermediate language called Java bytecodes. The bytecodes is the platform-independent code interpreted by the Java interpreter. With an interpreter, each Java bytecodes instruction is parsed and run on the computer.

---

[4] A paying plug-in exist for Netscape Navigator

[5] For more details about the Java Platform, See "The Java Tutorial", *http://java.sun.com/docs/books/tutorial/index.html*

[6] Those words are explain in details by James Gosling and Henry McGilton "The Java Language Environment", *http://java.sun.com/docs/white/langenv/*

Compilation happens just once; interpretation occurs each time the program is executed. The figure 6-3 illustrates how this works.

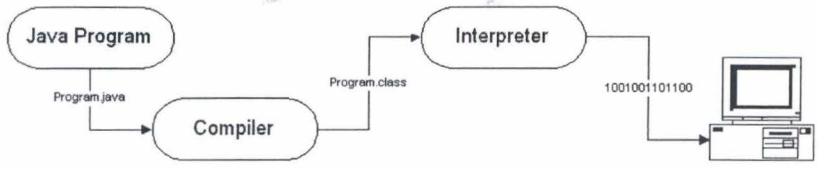


Figure 6-3. *Difference between Compilation and Interpretation in Java*

Java bytecodes can be seen as the machine code instructions for the JVM. Every Java interpreter, whether it's a Java development tool or a web browser that can run Java applets, is an implementation of the JVM. Java bytecodes help make "write once, run anywhere" possible. Java programs can be compiled into bytecodes on any platform that has a Java compiler. The bytecodes can then be run on any implementation of the JVM (figure 6-4).

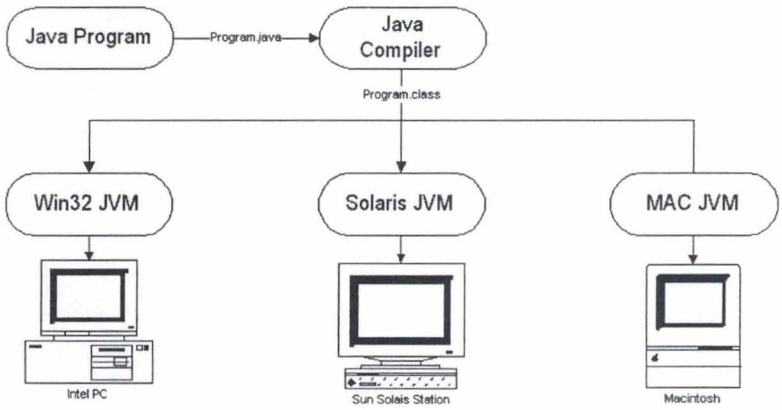


Figure 6-4. *The cross-platform functionality of Java*

The Java code is compiled into bytecodes by the JDK[7]. Some IDE[8] exists to make Java programming easier. We will use such a tool, namely Borland's JBuilder 2.0.

3.3.2. Java applet

A *Java applet* is a Java program that adheres to certain conventions that allows it to run within a Java-enabled browser such as Netscape Navigator or Internet Explorer. Unlike an ActiveX control that is downloaded just one time, a Java applet is downloaded every time it's needed.

3.3.3. Swing

Swing[9] is a Java API[10] that simplifies and streamlines the development of UI components. The Swing components are the visual components (such as menus, tool bars, dialogs...) that are used in graphically based applets and applications[11].

---

[7] Java Development Kit
[8] Integrated Development Environment
[9] officially known as Java Foundation Class
[10] Application Programming Interface
[11] a list of the swing component is provided in the appendix A section 8.5.

One the most important feature of Swing components is that they are lightweight [Sun99a]. That means they don't use any platform-specific implementations. Instead, Swing creates its components using pluggable look-and-feel (L&F) modules that are written from scratch and don't use any peer code at all. Consequently, Swing components can typically be incorporated into a program using less code that older "heavyweight" components required[12]. Therefore, Swing components use fewer system resources and produce smaller and more efficient applications than their heavyweight AWT counterparts. Swing has three standard L&F: Metal, Window and Motif.

3.3.4. Java Plug In

If the use of Swing components will enhance greatly our application interfaces, it is a also an issue. Even if it is now part of the Java 2 platform (aka JDK 1.2), it is unfortunately not fully supported by the current versions of the web browsers. In order to execute an applet that uses Swing components, the browser has to be upgraded with the adequate plug-in.

A *plug-in* is an implementation of the part of a browser that dynamically loads when needed [Friesen99]. Both Netscape and Microsoft have added this feature to their browsers.

Sun's solution to the fact that the current browsers don't support the Swing classes is the creation of a plug-in for Java, known as *Java Plug-in.*

3.4. The UPE high-level components

During our stay at the University of Port Elizabeth (South Africa), we designed a library of High Level Components[13] using Swing. What we decided to call the *High Level Components* are *final components* opposed to the *Low Level Components*, which are *non-final components*. That means that the user can include them directly into a form. They have simple methods and properties that make them easy to use.

When working with High Level Components, the user will have to write minimal code to finally get complete and useful components. The High Level components are the combination of Low Level Components, which means that the entire interaction between those components is already implemented inside the High Level Components. This library will be used to design our applications.

4. Middleware

A middle tier is something that helps connecting one endpoint to another (an applet to a database, for example) and along the way adds a little something of its own [Hunter98, p.244]. The most compelling reason for putting a middle tier between a client and a data source is that software in the middle tier (commonly referred to as middleware) can include business logic. *Business logic* abstracts complicated low-level tasks (such as updating database tables) into high-level tasks (placing an order), making the whole operation simpler and safer.

In our case, the middleware will be used for several applications.

[12] a more detailed description of the Swing architecture is provided in appendix A section 4.2.1.3.
[13] more details about the development of this library are available in appendix A

First, the middleware should provide a way to generate or modify web pages. This will be used in the following situations:

- When the user requests a dynamic page which must be generated from data stored in the database.
- When the user requests a static page whose navigation will be placed dynamically.

Secondly, the middleware should provide a way to access directly the IOs database. This will be used in the following situations:

- When the teacher wants to update the IOs database.
- When an applet needs some information about IOs stored in the database.

4.1. Dynamic pages generation

At this stage we have to find a technology able to generate and modify web pages dynamically. Lots of products on the market provide that feature and in such a fast moving market, new technologies appears every month.

The implementation of the VESALE project has not started yet and consequently, we don't want our programs to be too dependent of a particular technology.

4.1.1. CGI

The *Common Gateway Interface*, normally referred to as CGI, was one of the first practical techniques for creating dynamic content [Hunter98, p.2]. With CGI, a web server passes certain requests to an external program. The output of this program is then sent to the client in place of a static file. The advent of CGI made it possible to implement all sorts of new functionality in web pages, and CGI quickly became a de facto standard, implemented on dozens of web servers.

It's interesting to note that the ability of CGI programs to create dynamic web pages is a side effect of its intended purpose: to define a standard method for an information server to talk with external applications. This origin explains why CGI has perhaps the worst life cycle imaginable. When a server receives a request that accesses a CGI program, it must create a new process to run the CGI program and then pass to it, via environment variables and standard input, every bit of information that might be necessary to generate a response. Creating a process for every such request requires time and significant server resources, which limits the number of requests a server can handle concurrently. The figure 6-5 illustrates how a process is created for each request.

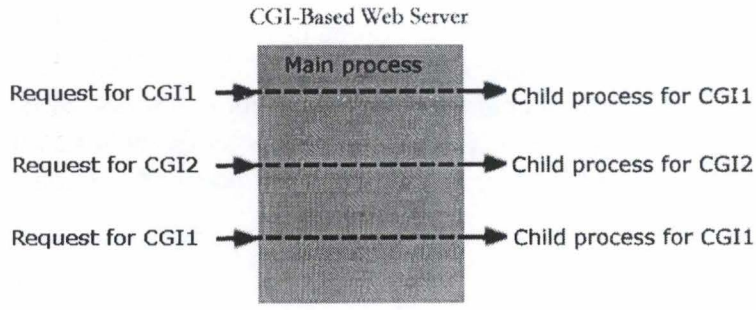


Figure 6-5. *The CGI life cycle*

Even though the FASTCGI technology alternative improves this process proliferation problem by creating a single persistent process for each FastCGI program, it is still resources consuming.

Another problem with this CGI technology is that it does nothing to help a CGI program to interact more closely with the server.

4.1.2. Server specific technologies

A few others server specific technologies are available in order to create dynamic content:

- **Server extension APIs**

Several companies have created proprietary server extension APIs for their web servers [Hunter98, p.4]. For example, Netscape provides an internal API called NSAPI (now becoming WAI) and Microsoft provides ISAPI. Using one of these APIs, it is possible to write server extensions that enhance or change the base functionality of the server, allowing the server to handle tasks that were once relegated to external CGI programs. As the figure 6-6 illustrates, server extensions exist within the main process of a web server.

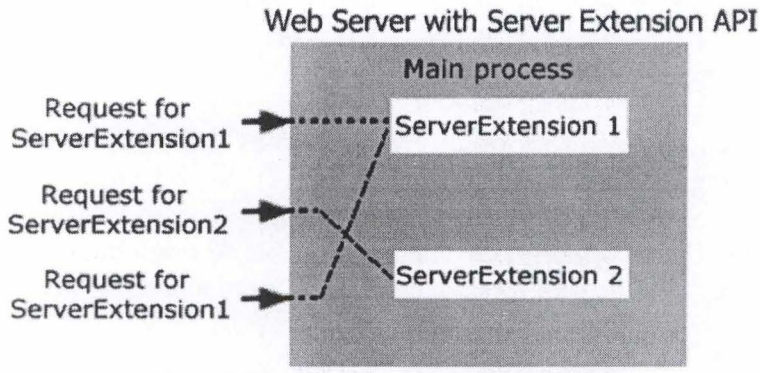


Figure 6-6. *The Server Extension life cycle*

Because server-specific APIs use linked C or C++ code, server extensions can run extremely fast and make full use of the server's resources. Server extensions, however, are not a perfect solution by any means. Besides being difficult to develop and maintain, they pose significant security and reliability hazards: a crashed server extension can bring down the entire server. And, of course, proprietary server extensions are inextricably tied to the server API for which they were written and often tied to a particular operating system as well.

- **Active Server Pages**

Microsoft has developed a technique for generating dynamic web content called *Active Server Pages*, or sometimes just ASP [Hunter98, p.5]. With ASP, an HTML page on the web server can contain snippets of embedded code (usually VBScript or JScript, although it's possible to use nearly any language). This code is read and executed by the web server before it sends the page to the client. ASP is optimised for generating small portions of dynamic content.

- **Server-side JavaScript**

Netscape too has a technique for server-side scripting, which it calls *server-side JavaScript*, or SSJS for short [Hunter98, p.5]. Like ASP, SSJS allows snippets of code to be embedded in HTML pages to generate dynamic web content. The difference is that SSJS uses JavaScript as the scripting language. With SSJS, web pages are precompiled to improve performance. Support for server-side JavaScript is available only with Netscape FastTrack Server and Enterprise Server Version 2.0 and above.

Although all these technologies are among the most efficient for creating dynamic content, they are all tied to a specific web server. At this stage of the VESALE project, no decisions have been made concerning the web server used. Consequently, it would be injudicious to opt for one of these technologies considering that the switch from a server to another would require the rewriting of the whole code.

4.1.3. Java servlets

On the web, middle tiers are often implemented using Java servlets [Hunter98, p.6]. A servlet is a generic server extension (a Java class) that can be loaded dynamically to expand the functionality of a server. They provide a convenient way to connect clients built using HTML forms or applets to back-end servers. A client communicates its requirements to the servlet using HTTP, and the business logic in the servlet handles the request by connecting to the back-end database. Moreover it provides several techniques to generate dynamic web pages.

4.1.3.1. Servlets life cycle

Servlets are commonly used with web servers, where they can take the place of CGI scripts. A servlet is similar to a proprietary server extension, except that it runs inside a JVM on the server (figure 6-7), so it is safe and portable. Servlets operate solely within the domain of the server: unlike applets, they do not require support for Java in the web browser.

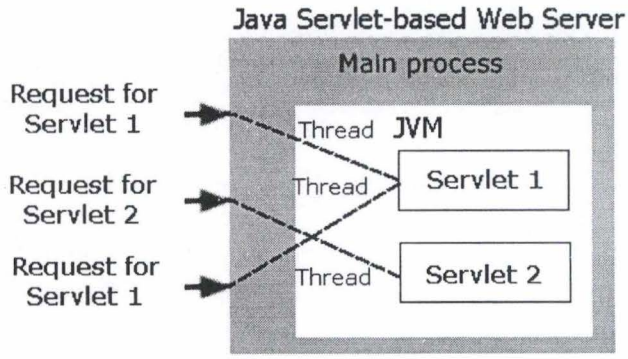


Figure 6-7. *The servlet life cycle*

Unlike CGI and FastCGI, which use multiple processes to handle separate programs and/or separate requests, separate threads within the web server process handle all servlets. This means that servlets are also efficient and scalable. Because servlets run within the web server, they can interact very closely with it to do things that are not possible with CGI scripts.

Another advantage of servlets is that they are portable: both across operating systems as we are used to with Java and also across web servers. All of the major web servers support them.

4.1.3.2. Servlets features

We believe that servlets offer a number of advantages over other approaches, including portability, power, efficiency, endurance, safety, elegance, integration, extensibility, and flexibility. Let's examine each in turn [Hunter98, p.10].

- **Portability**

Because servlets are written in Java and conform to a well-defined and widely accepted API, they are highly portable across operating systems and across server implementations. It is possible to develop a servlet on a Windows NT machine running the Java Web Server and later deploy it effortlessly on a high-end Unix server running Apache.

- **Power**

Servlets can harness the full power of the core Java APIs: networking and URL access, multithreading, image manipulation, data compression, database connectivity, internationalisation, remote method invocation (RMI), CORBA connectivity, and object serialisation, among others.

Servlets are also well suited for enabling client/server communication. With a Java-based applet and a Java-based Servlet, object serialisation is used to handle client/server communication quite easily. This object serialisation feature allows an applet and a servlet to exchange Java objects without having to develop a custom protocol the handle the communication.

- **Efficiency and endurance**

Servlet invocation is highly efficient. Once a servlet is loaded, it generally remains in the server's memory as a single object instance. Thereafter, the server invokes the servlet to handle a request using a simple, lightweight method invocation. Unlike with CGI, there's no process to spawn or interpreter to invoke, so the servlet can begin handling the request almost immediately. Multiple, concurrent requests are handled by separate threads, so servlets are highly scalable.

Servlets, in general, are naturally enduring objects. Because a servlet stays in the server's memory as a single object instance, it automatically maintains its state and can hold on to external resources, such as database connections, that may otherwise take several seconds to establish.

- **Safety**

Servlets support safe programming practices on a number of levels. Because they are written in Java, servlets inherit the strong type safety of the Java language. In addition, the servlet API is implemented to be type-safe. While most values in a CGI program, including a numeric item like a server port number, are treated as strings, values are manipulated by the Servlet API using their native types, so a server port number is represented as an integer. Java's automatic garbage collection and lack of pointers mean that servlets are generally safe

from memory management problems like dangling pointers, invalid pointer references, and memory leaks.

Servlets can handle errors safely, due to Java's exception-handling mechanism. If a servlet divides by zero or performs some other illegal operation, it throws an exception that can be safely caught and handled by the server, which can politely log the error and apologise to the user. If a C++-based server extension were to make the same mistake, it could potentially crash the server.

- **Elegance**

The elegance of servlet code is striking. Servlet code is clean, object oriented, modular, and amazingly simple. One reason for this simplicity is the Servlet API itself, which includes methods and classes to handle many of the routine chores of servlet development. Even advanced operations, like cookie handling and session tracking, are abstracted into convenient classes.

- **Integration**

Servlets are tightly integrated with the server. This integration allows a servlet to co-operate with the server in ways that a CGI program cannot. For example, a servlet can use the server to translate file paths, perform logging, check authorisation, and, in some cases, even add users to the server's user database. Server-specific extensions can do much of this, but the process is usually much more complex and error-prone.

- **Flexibility**

Servlets are also quite flexible. For instance, a HTTP Servlet can be used to generate a complete web page using lots of different methods.

4.1.3.3. HTTP request handling

When a client connects to a server and makes an HTTP request, the request can be of several different types, called methods [Hunter98, p.14]. The most frequently used methods are GET and POST. Put simply, the GET method is designed for getting information, while the POST method is designed for posting information.

The GET method, although it's designed for reading information, can include as part of the request some of its own information that better describes what to get, such as an x, y scale for a dynamically created chart. This information is passed as a sequence of characters appended to the request URL in what's called a query string. Placing the extra information in the URL in this way allows the page to be bookmarked or emailed like any other.

The POST method uses a different technique to send information to the server because in some cases it may needs to send megabytes of information. A POST request passes all its data, of unlimited length, directly over the socket connection as part of its HTTP request body. The exchange is invisible to the client. The URL doesn't change at all. Consequently, POST requests cannot be bookmarked or emailed or, in some cases, even reloaded.

The servlet instance loaded inside the memory of the web server simply needs to define a doGet() function to be able to respond to GET request and a doPost() function to be able to respond to POST requests. The response to a specific HTTP request returned by the web server is the value returned by either the doGet or doPost functions. The result can be either an HTML page, a string, a Java serializable object, a picture or any kind of media. The figure 6-7 illustrates how the web server handles an HTTP request.

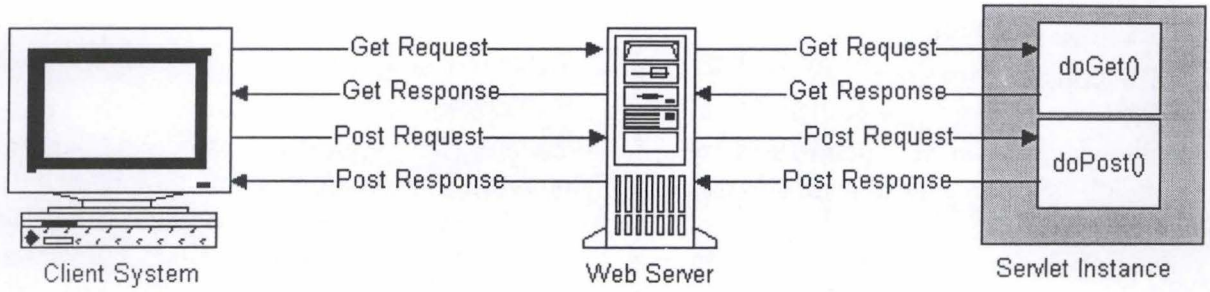


Figure 6-7. *An HTTP servlet handling GET and Post request*

4.1.3.4. Pages generation using servlets

The Servlet API provides various ways to generate dynamic web pages. The main methods are the following:

- **Server-side includes**

Servlets can be embedded inside HTML pages with something called server-side include (SSI) functionality [Hunter98, p.27]. In many servers that support servlets, a page can be pre-processed by the server to include output from servlets at certain points inside the page.

A server that supports SSI detects the <SERVLET> tag in the process of returning the page and substitutes in its place the output from the Servlet (figure 6-8).

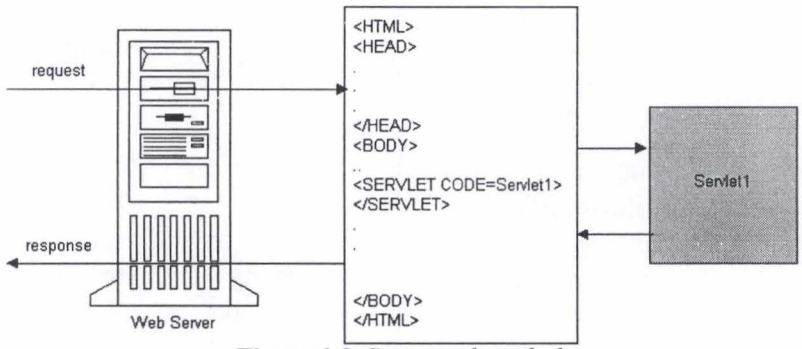


Figure 6-8. *Server-side include*

- **Servlet chaining**

In many servers that support servlets, a request can be handled by a sequence of servlets [Hunter98, p.30]. The request from the client browser is sent to the first servlet in the chain. The response from the last servlet in the chain is returned to the browser. In between, the out-

put from each servlet is passed (piped) as input to the next servlet, so each servlet in the chain has the option to change or extend the content (figure 6-9).

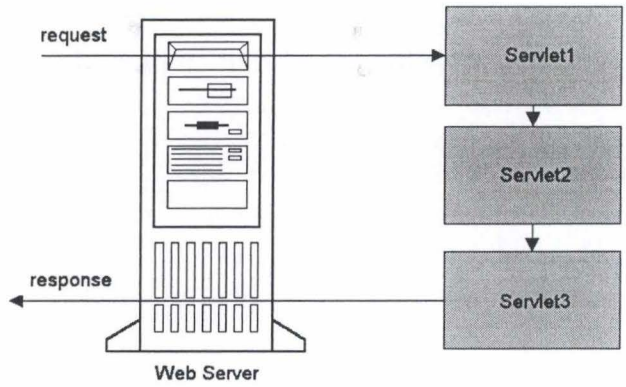


***Figure 6-9**. Servlet chaining*

Servlet chaining can change the way web content creation is approached. For instance, a site can be improved by replacing automatically some custom tags on every page served by the web server.

– **Java Server Pages**

Recently, Sun announced a new way to use servlets, called Java Server Pages (JSP) [Hunter98, p.37]. JSP's functionality and syntax bear a remarkable resemblance to ASP (4.1.2.).

JSP operates in many ways like server-side includes. The main difference is that instead of embedding a <SERVLET> tag in an HTML page, JSP embeds actual snippets of servlet code (figure 6-10). It's an attempt by Sun to separate content from presentation, more convenient than server-side includes for pages that have chunks of dynamic content intermingled with static content in several different places.

Just like server-side includes and servlet chaining, JSP doesn't require any changes to the Servlet API. But it does require special support in the web server. This support is not yet included in most web servers but it's expected to be introduced soon.

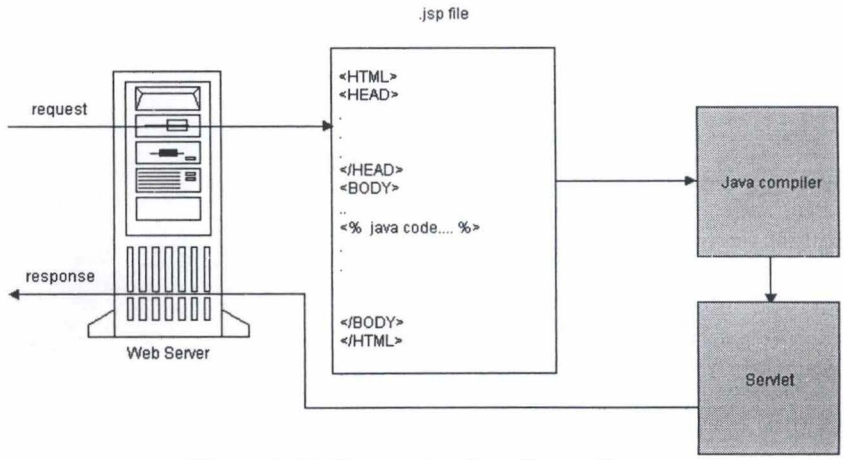


***Figure 6-10**. Generating Java Server Pages*

Unfortunately, the JSP technology is still too young and therefore too unstable to be used to generate pages.

The SSI technology is efficient and implemented in most web servers supporting servlets but can be not very efficient in certain applications. For instance, if the result of a same database query has to be placed at several scattered places on a same web page, the query will be executed as many times as the <SERVLET> tag is included in the page as shown in the figure 6-11. Consequently, this would not be the best technology to use in order to generate our web pages from data stored in the IO database.

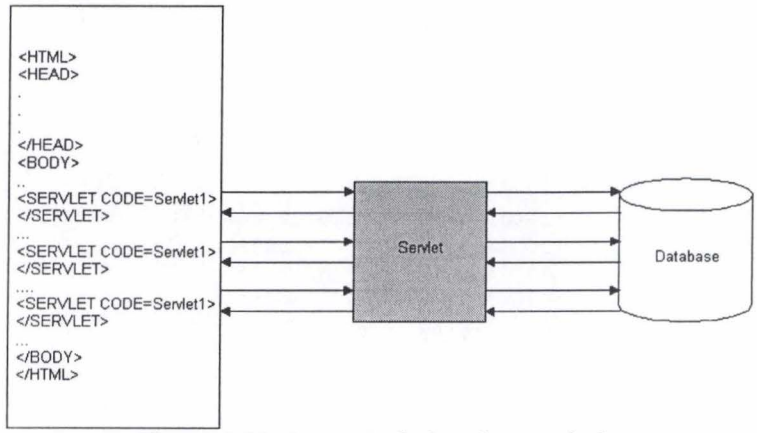


Figure 6-11. *A page including three includes*

The servlet chaining technology would certainly be the most efficient considering our specific needs, namely generating a dynamic page from data stored in the database and adding dynamically the navigation features to the pages served. For instance, we could add a servlet in the chain whose work is to add the navigation features. Identically, the dynamic pages could be generated by passing them to a particular servlet in the chain whose job would be to replace specific tags by the content of a single database query.

4.1.4. Pages generation technology choice

From all this, it appears that the servlet technology is the most adapted to our specific requirements. This Java-based architecture that we opted for presents the advantage of being *platform independent.*

In order to generate and modify the web pages served, we will use mainly the servlet chaining method.

4.2. Database connectivity : the JDBC API

Now that we have opted for the servlets technology on the middleware, we need a method to communicate with a database using the Java programming language. The solution provided by Java is called the JDBC[14] API.

[14] Java DataBase Connectivity

JDBC is an SQL[15] level API, which means that it allows the construction of SQL statements and their embedding inside Java API calls [Reese97, p.51]. In short, programmers basically use SQL. JDBC lets programmers smoothly translate between the world of the database and the world of the Java application. The results from the database, for instance, are returned as Java variables, and access problems get thrown as exceptions. JDBC attempts to remain as simple as possible while providing developers with maximum flexibility.

If a Java program wants to communicate with a particular DBMS, it must register its specific JDBC driver. As shown in the figure 6-12, a Java program can communicate with several DBMSs.

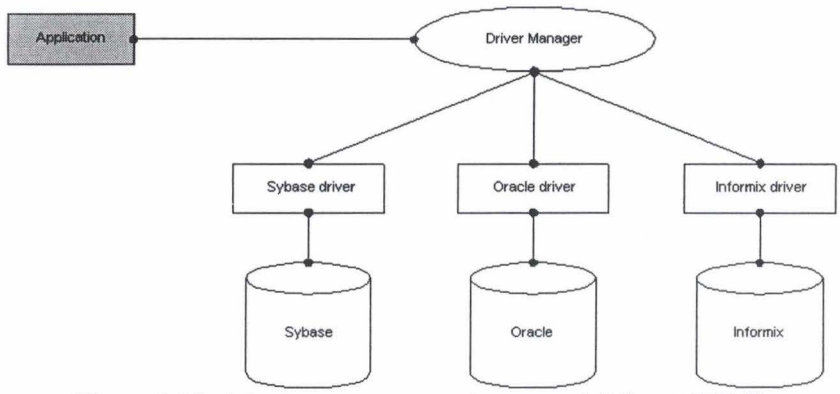


***Figure 6-12**. A Java program accessing several different DBMSs*

Another very interesting feature of JDBC is that a programmer can develop an application with a specific DBMS and decide suddenly to use another DBMS. If the new DBMS offers a JDBC driver, the programmer won't have to modify his program to take this change into account.

4.3. Java Web Server

The next step is to choose a web server supporting Java servlets. We opted for Sun's Java Web Server, unofficially considered the reference implementation for how a servlet engine should support servlets [Hunter98, p.8]. It is written entirely in Java and is very easy to install and configure.

The choice of a web server is not a crucial one at the moment as we opted for a platform independent technology, namely the Java servlets. If in later stages of the VESALE project it appears that this web server is not suitable anymore, the servlets written could be instantaneously plugged into another web server.

5. DBMS

The choice of a DBMS to implement our IOs database was in fact a very simple one as it had already been decided to use Oracle 8 to implement all the VESALE databases. As Oracle 8 provides a JDBC driver, it will be easy to make queries to the database through a Java program using the JDBC API.

[15] Standard Query Language

6. Conclusion

In this chapter, we have researched technologies and methods that will be used to develop our part of the VESALE project. Those technologies being *platform independent*, the elements that we will implement could be easily adapted for the technologies chosen for the VESALE project.

In the next chapter, we will use those technologies to implement the elements described in the analysis.

7

Design

1. Introduction

In this chapter, we will present the design process that we followed in order to implement the elements described in chapter 5 using the technologies chosen in chapter 6.

First, we will describe the page template that we created for the course pages. Then, we will explain how we created the IOs database and how we developed tools allowing the user to manipulate it. Afterwards, we will explain how we generated dynamically both the navigation and the IOs description pages. Thereafter, we will describe the interactive applications we developed. Finally, we will explain how we have rewritten the course to adapt it for the web.

2. Page design

In this section, we will illustrate how we have designed a page template that will be used repeatedly throughout the course.

2.1. VESALE logo

The logo that will be displayed on every pages[1] must be simple and effective in its design. It must be different from the University's logo because it links to the VESALE Home Page. Furthermore, the logo must convey a certain symbolic of what VESALE is.

[1] See chapter 5 section 5.3.2.

After some preliminary tests, we have realised three different logo concepts:

Figure 7-1. First logo

Our first logo (figure 7-1) represents a student juggling with concepts. This logo has two major drawbacks: it is difficult to identify the student at a lower scale and its general aspect is a little too "childish".

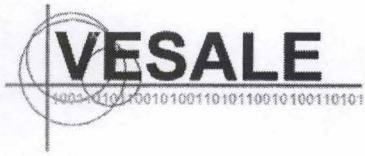


Figure 7-2. Second logo

Our second logo (figure 7-2) symbolises the human thinking with circles and the computer reasoning with lines and bits. It also has drawbacks: it is not sober and totally unscalable since details disappear at a lower scale.

Figure 7-3. Third logo

Our third logo (figure 7-3) is simple, effective and scalable. The shape of the triple V reminds the support of the course, namely the World Wide Web (WWW). The black V represents the interaction between human thinking (V crossed by circles) and the machine (V crossed by squares).

We have decided to use the third logo because we think that it is the one who will fit better on our pages and that convey the strongest symbolic.

2.2. Page structure

In order to design the page layout grid presented in the analysis[2] using HTML, we have chosen to use HTML tables.

One of the advantages of tables for designing layout grids is that it has not to be symmetrical. For instance, a table can have two columns but the left one can be composed of one row while the right one has two (figure 7-4).

[2] See chapter 5 section 5.3.2.

Cell 1	Cell 2
	Cell 3

```
<table border="1" width="75%">
 <tr>
  <td rowspan="2">Cell 1</td>
  <td>Cell 2</td>
 </tr>
 <tr>
  <td>Cell 3</td>
 </tr>
</table>
```

***Figure 7-4**..A simple table and the corresponding HTML code*

Moreover, a table cell can contain another table.

We used *asymmetrical* and *embedded* tables to design our layout grid. The figure 7-5 shows the general table design of the layout.

Logo	Chapter's Name Page Name	Sequential Navigation Zone
Chapter Structure Ad Hoc Navigation	Text Zone	
	Copyright Information	Sequential Navigation Zone

***Figure 7-5**. The general table structure of the layout grid*

Since tables are also used to place graphical elements at specific positions, the table structure we designed is far more complicated that the one presented in figure 7-5. The final look of the layout grid including all the graphical elements is presented in figure 7-6[3].

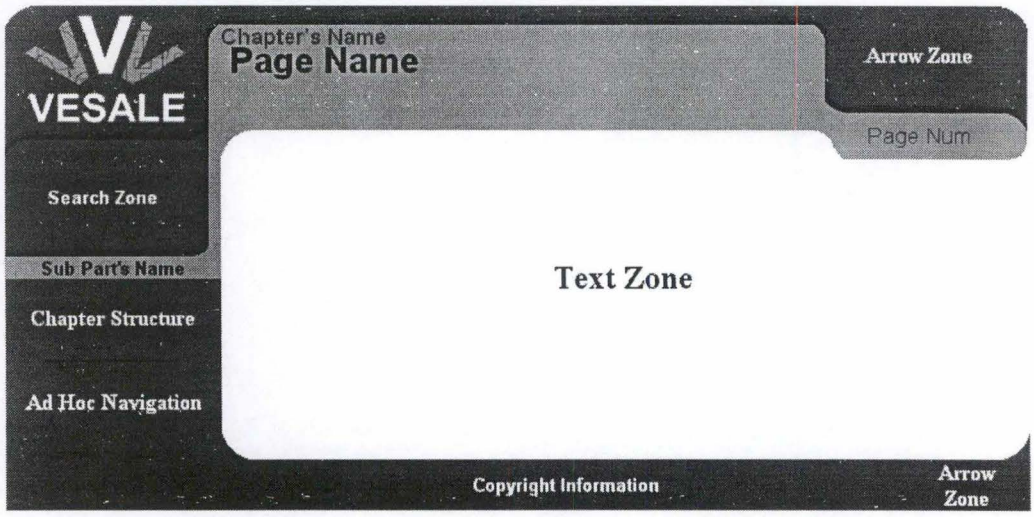


***Figure 7-6.** The final layout grid including graphical elements*

[3] The corresponding code is in appendix D section 1.

2.3. Cascading Style Sheet

The *cascading style sheet*[4] that we have designed is an external file called by every pages of the site. Therefore, if we want to change the style of a specific element in every pages, all we have to do is to modify this single file[5].

The tags we used for the navigation elements are subclasses of existing tags. For instance, the left column links are subclasses of the "anchor" HTML element. This allows the designer of the page content to use standard tags (e.g. P, H1 to H6) in order to include content in the page body.

2.4. Navigation scenario

As presented in the analysis[6], there are multiple ways of accessing a specific page in the course. Therefore, a page will provide a way of accessing other pages according to the hierarchical and sequential architecture.

2.4.1. Sequential navigation

An arrow zone is displayed on every pages (figure 7-7). By simply clicking on the left arrow, the previous page in the sequential architecture[7] will be displayed. Identically, by clicking on the right arrow, it is the next page in this architecture that will be displayed.

Moreover, when the user rolls the mouse over one arrow, a hint will indicate which concept is presented in the linked page. The page order in the sequential architecture of the chapter is displayed below the arrows.

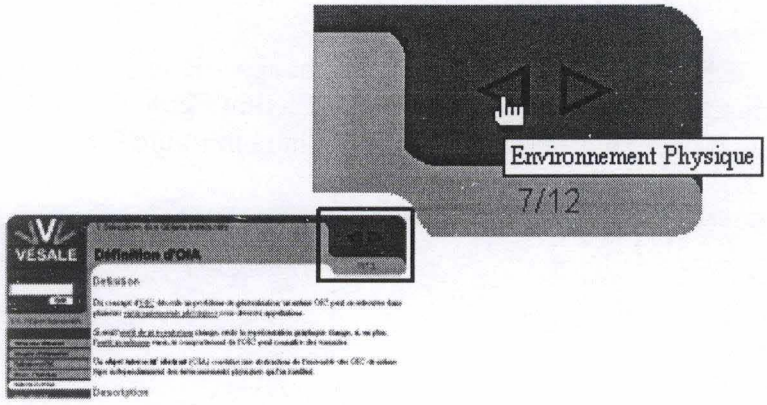


Figure 7-7. *The arrow zone*

2.4.2. Hierarchical navigation

The left column shows the structure of the current section to the user (figure 7-8). In that structure, the current sub-section of the page in the hierarchical architecture[8] is highlighted with a white background.

---

[4] See chapter 6 section 3.2.
[5] The style sheet code is in appendix D section 2.
[6] See chapter 5 section 5.2.
[7] See chapter 5 section 5.2.2.
[8] See chapter 5 section 5.2.1.

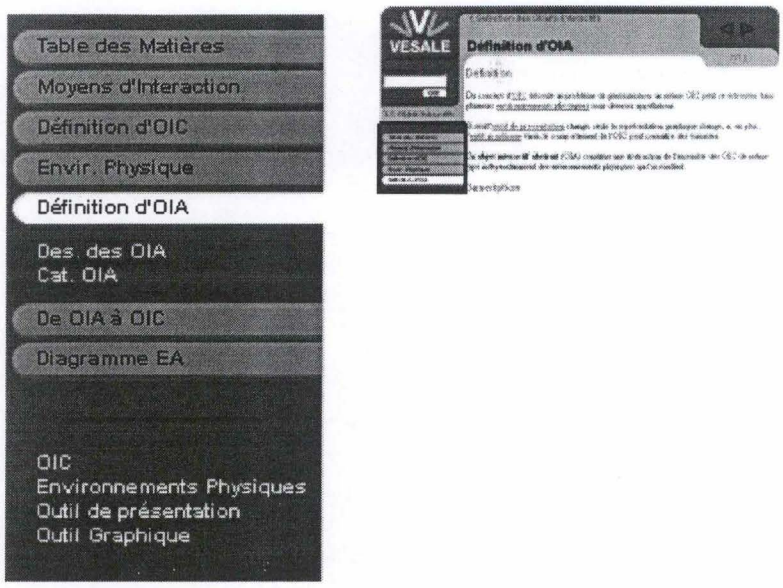


***Figure 7-8**. The hierarchical structure zone*

Moreover, the pages provide links to both the chapter and section first pages. As shown in figure 7-9, the zone 1 gives the names of the current page and chapter. The chapter's name is preceded by it's position in the course structure and links to the chapter's first page. The text in zone 2 is the name of the chapter's section containing the current page. It is preceded by its order in the chapter structure and links to the section first page. Finally, the VESALE logo in zone 3 links to the VESALE Home Page.

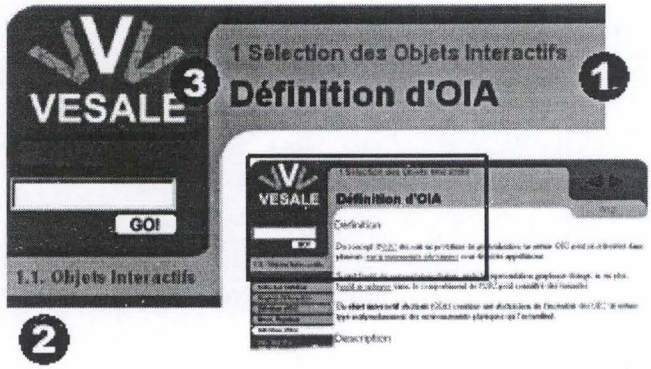


***Figure 7-9**. The user can go back to the chapter's first page (1) or to the section's first page (2). He can also access the site home page by clicking on the VESALE logo (3).*

2.4.3. Conceptual navigation

A dedicated place in the left column is reserved for a copy of the embedded links (figure 7-8 above).

3. IOs database

3.1. Database physical schema

The physical schema of the IOs database[9] has been derived from the conceptual schema[10] (figure 7-10).

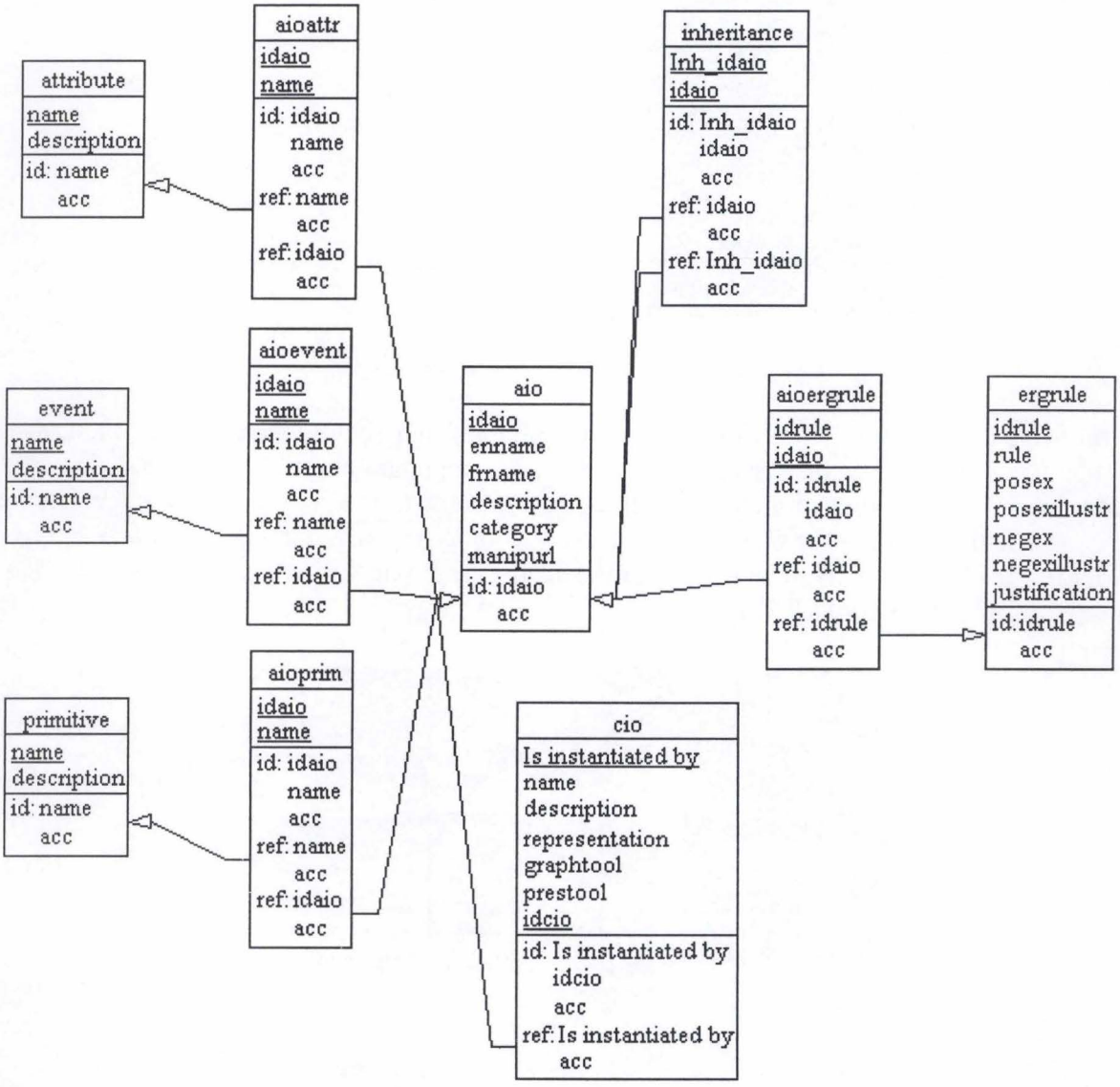


***Figure 7-10**. The IOs database physical schema*

The most important choice we have made in the design of the database is to store all the graphics as a string representing the picture's URL rather than as a binary object. First, strings are much easier to handle that binary objects. This decision will save us a lot of time when exchanging pictures between HTML pages, servlets and the database. Secondly, the pictures stored in the database will be mainly displayed inside HTML pages, which means that each picture needs an URL to be embedded inside a page. It is much more efficient to insert the

[9] The SQL script used to generate the database is in the appendix B section 2.
[10] See appendix B section 1.

URL of the picture file inside the page rather than having a servlet sending a binary stream representing the picture to the user's browser.

3.2. Database manipulation tools

3.2.1. HTML forms

The next stage is to provide the tools that will be used by the teacher in order to manipulate the database. We decided to implement those tools as web pages using forms. HTML forms are much easier to create and more efficient to use than Java applets for instance. They provide the basic interaction objects that we need such as edit boxes, drop–down combo boxes, list boxes and multi-line edit boxes.

3.2.2. Request handling process

The HTML forms used to manipulate the database are linked with servlets in order to execute database queries. The request handling process is illustrated in figure 7-11.

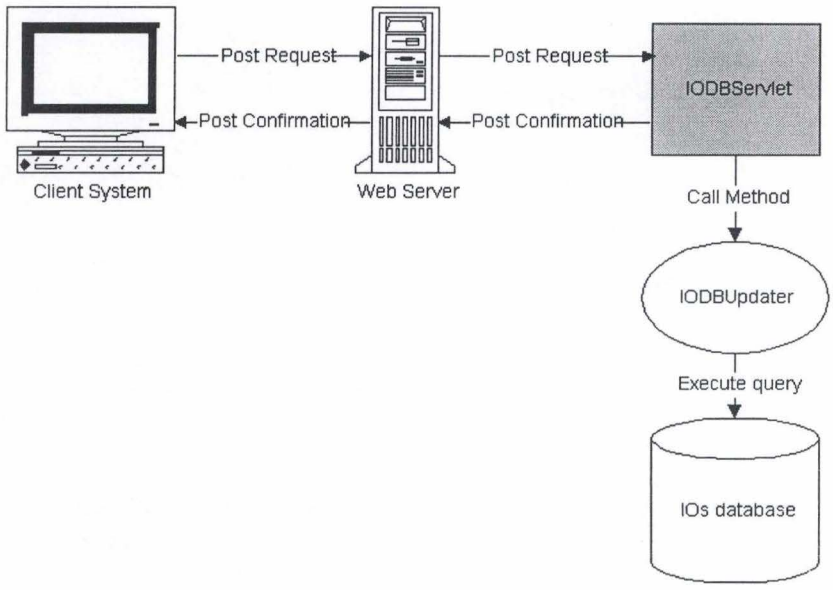


Figure 7-11. *The request handling process*

When the user has fill in a form whose function is to update the database, he pushes on the submitting button of the form. The result of this action is that a POST HTTP request will be send to a particular servlet on the web server along with all the data entered by the user. This servlet (*IODBServlet*) will gather all the data received and send it to the *IODBUpdater* object by calling a database modification primitive (figure 7-11). This *IODBUpdater* object contains what we would call the *business logic* of our application. For instance, if we want to change completely the database, we just have to modify this object without having to update the HTML forms and the *IODBServlet*.

3.3. Database content

Now that the database is created and that we have tools to update it, we have to fill it with information concerning IOs. As part of this thesis, we will limit ourselves to a few demonstration IOs that we will develop entirely.

The main references that we will use to add content are the following:

- Une description orientée objet des objets interactifs abstraits utilisés dans les interfaces homme-machine [Vanderdonckt96]
- GUI design essentials [Weinschenk97]
- GUI design handbook [Fowler98]
- Ameritech Graphical User Interface Standards and Design Guidelines [Ameritech99]
- MacOs Graphical User Interface Standards and Design Guidelines [Apple99]
- Java Look and Feel Design Guidelines [Sun99b]

4. Dynamic pages generation

As stated in chapter 6[11], we will use Java servlets to generate both the navigation elements and the IOs description pages.

4.1. Dynamic navigation

As stated in chapter 5[12], we will generate all the navigation elements of the pages dynamically. The idea is to provide the designers of the course syllabus with tools that give him great flexibility concerning the course structure.

In a dynamic navigation generation, the course structure will be stored somewhere on the web server or in a database. If the teacher wants to modify the course structure, he will only have to modify the stored course structure.

The implementation of several *pedagogical scenarios* as described in chapter 5[13] would require the management of several course structures descriptions. However, we have decided by simplicity to limit ourselves to the case of a unique course structure.

4.1.1. Course structure

To implement this dynamic navigation generation feature, we have first developed a way of describing the course structure. This could be implemented either as a text file stored on the web server or as part of the database.

The solution we opted for is the text file format. The course structure will be described in a text file following a specific syntax[14].

[11] See chapter 6 section 4.
[12] See chapter 5 section 5.4.
[13] See chapter 5 section 4.2.
[14] The course structure syntax is in appendix E section 1 and the course definition is in appendix E section 5.

4.1.2. Custom tag

Every static web pages on the server as well as the dynamic pages contain parts that have to be replaced with the HTML code implementing the navigation elements. Those parts are identified by a specific tag indicating which element must replace it.

This tag has the following format: <VESALE=*replacedelement*>, where *replacedelement* indicates which element has to be placed in the page[15].

4.1.3. Servlet chain

As stated in chapter 6[16], the method we will use to modify web pages dynamically is the servlets chaining one. The servlet chain implemented is shown in figure 7-12.

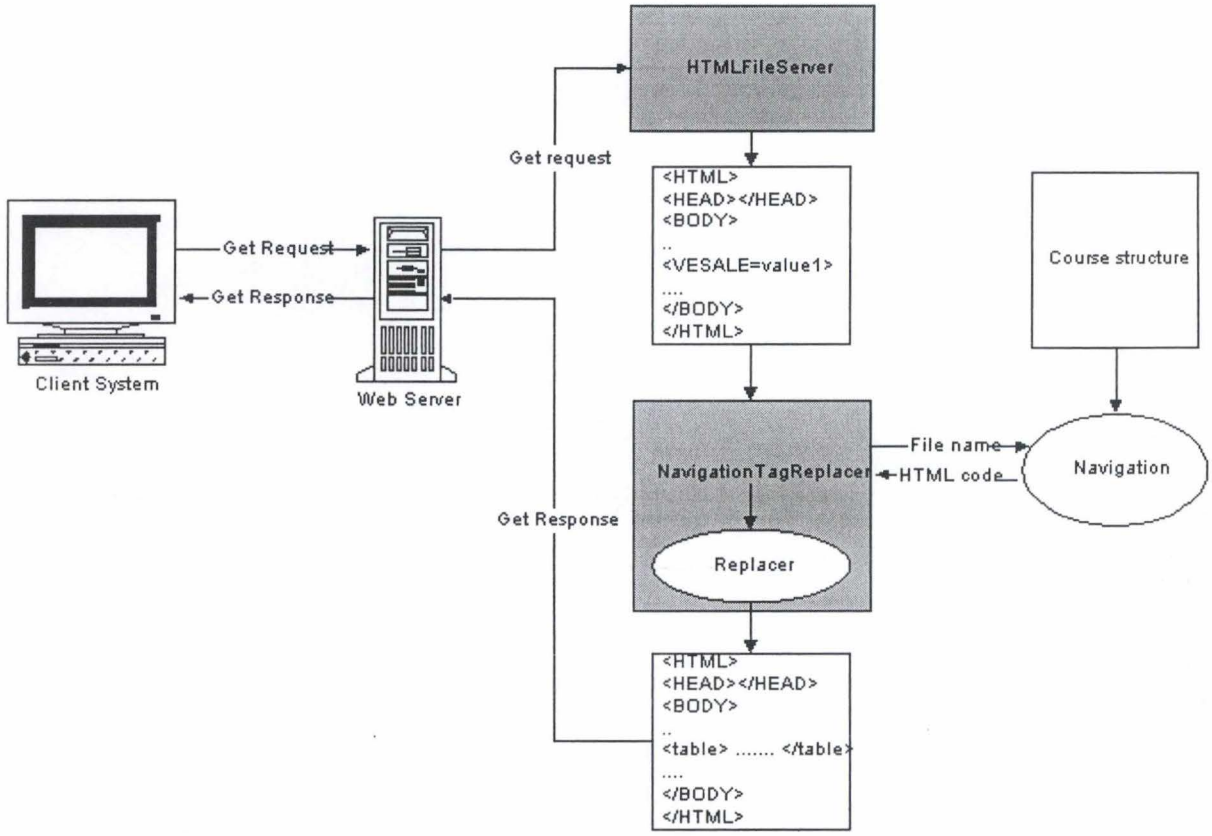


***Figure 7-12**. The dynamic navigation generation process*

The *HTMLFileServer* (figure 7-12) servlet is called by the web server when he receives a request for an .html file. This servlet analyses it and returns the requested file. This file contains the custom tags defined above and is then passed to the *NavigationTagReplacer* servlet.

The work of this *NavigationTagReplacer* servlet is to replace the custom tags with the HTML code implementing the navigation elements. This servlet will call some methods of the *Navigation* object in order to get this code.

[15] A pages containing those tags is displayed in appendix E section 2.1.
[16] See chapter 6 section 4.1.4.

This *Navigation*[17] object stores the course structure and contains the exact format of the navigation elements. As those elements are specific to each web page, those methods are called with the file name as an argument.
When the *NavigationTagReplacer* servlet knows the new content of all the tags that have to be replaced, the HTML code of the file is then passed to a *Replacer* object[18] which returns the same text in which all the replacements have been made. This new text[19] is then sent to the user's browser.

4.2. Dynamic pages

To generate web pages dynamically, we can easily use the same method that the one used to generate dynamically the navigation elements. A dynamic page would then be a static template stored on the web server which contains some custom tags at the places where the content must be inserted.

Instead of parsing each file twice to place the navigation elements and to add the dynamic content, it would be more efficient to make all the needed replacements at the same time. The figure 7-13 illustrates how we slightly modified our architecture.

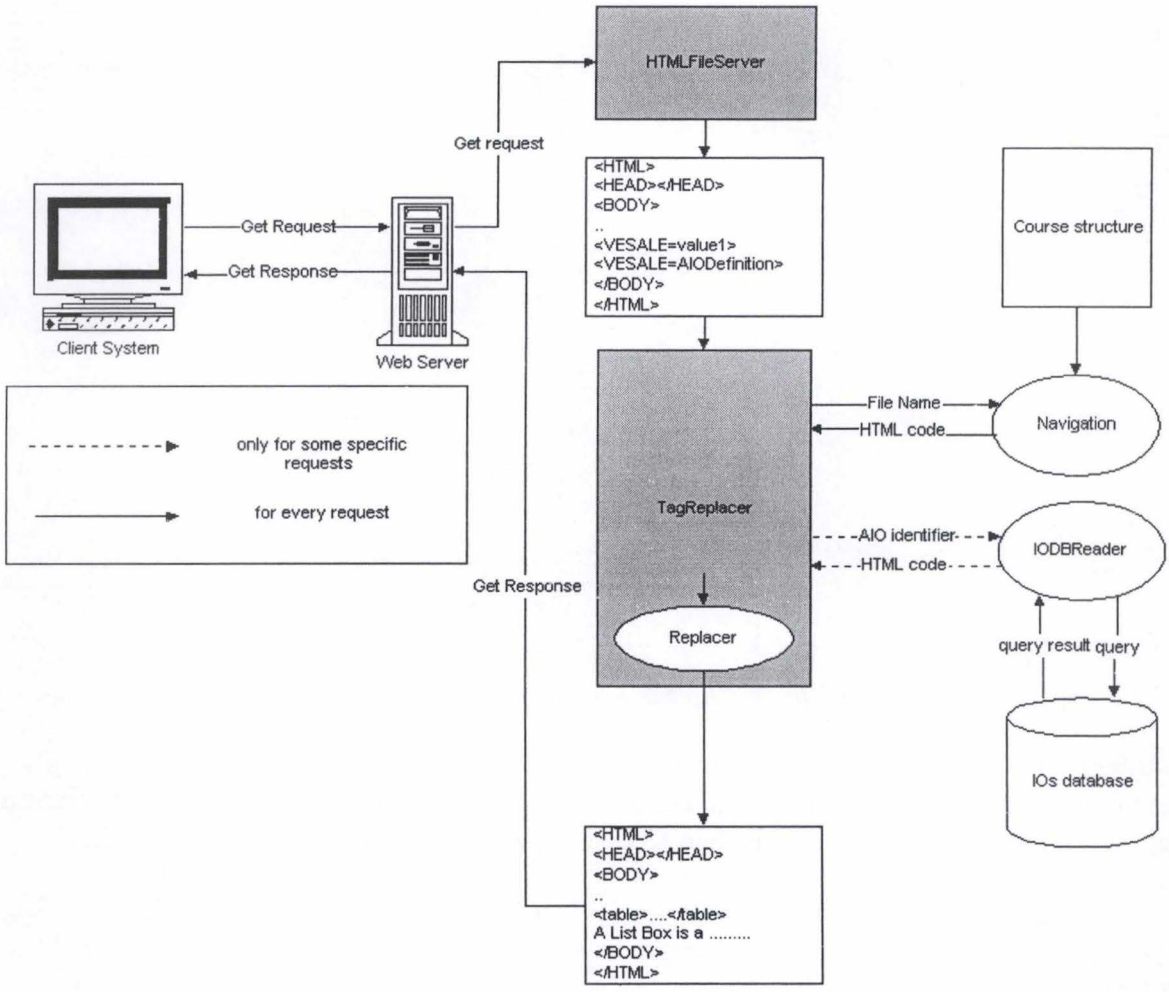


***Figure 7-13**. The dynamic pages and navigation generation process*

[17] The source code of the Navigation object is in appendix E section 3.1.
[18] The source code of the Replacer object is in appendix E section 3.2.
[19] A page containing the navigation elements is in appendix E section 2.2.

The *NavigationTagReplacer* servlet has been replaced by a more general *TagReplacer-Servlet*[20]. After having gathered the HTML code for the navigation elements, this servlet will check if the page contains dynamic content (using its file name that will use some conventions). If the page contains dynamic content concerning an AIO, the servlet will get the code for this content by calling some methods on the *IODBReader* object.

This *IODBReader*[21] object executes queries on the IOs database, formats them into HTML code and sends it to the Servlet.

Finally, all the replacements will be executed at the same time by the *Replacer* object and the new text will be send to the user's browser.

An example of generated IOs description pages is available in appendix E[22].

5. Interactive applications

In chapter 5[23], we have described the features that both applications should provide.

As *bootstrapping*[24] is the most important design principle of the VESALE project, we have followed the selection trees that we teach in order to select the IOs used in our applications interfaces[25].

Those applications have been developed as Java applets[26] using the UPE high-level components library[27].

5.1. IOs manipulation application

5.1.1. The UPE High-level components library

To develop this manipulation application, we needed a library of components that could be easily plugged into such an application. This choice was easy to make because the library that we have developed[28] fits perfectly in that purpose.

5.1.2. Application description

We have defined three zones that will each implement specific features:

- The AIO zone, which allows the user to select the AIO to manipulate
- The CIO zone, which allows the user to manipulate a CIO implementing the selected AIO
- The Attributes zone, which allows the user to manipulate indirectly the CIO

[20] The source code of the TagReplacer object is in appendix E section 3.4.
[21] The source code for the *IODBReader object is in appendix E section 3.3.*
[22] See appendix E section 4.
[23] See chapter 5 section 2.
[24] See chapter 2 section 1, chapter 3 section 3.7.
[25] This justification is in appendix F section 3.
[26] See chapter 6 section 3.3.2.
[27] See chapter 6 section 3.4.
[28] See chapter 6 section 3.4.

5.1.2.1. The AIO zone

The AIO zone allows the user to select a specific AIO. A CIO related to this AIO will be displayed in the CIO zone, and the user will be able to manipulate it (figure 7-14).

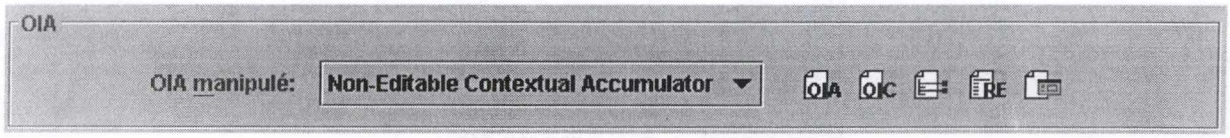


***Figure 7-14**. The AIO zone*

In order to implement the *related links* principle[29], a few links related to the manipulated AIO are provided. Those links are displayed as icons that react when the user rolls the mouse over it by displaying an hint indicating the destination of the link (figure 7-15).

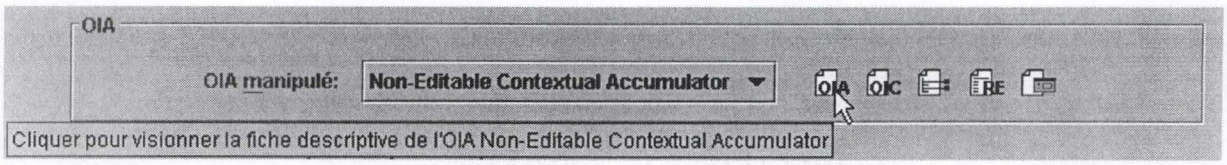


***Figure 7-15**. An hint is displayed when the student rolls the mouse over the icons*

The buttons on the right-hand side of the zone provide links to

- The description of the AIO
- The CIOs related to the displayed AIO
- The selection rules for the displayed AIO
- The ergonomic rules for the displayed AIO
- The cases in which the displayed AIO appears

5.1.2.2. The CIO zone

The CIO zone implements the *direct manipulation* principle[30]. This zone allows the user to manipulate directly a CIO related to the selected AIO.

The figure 7-16 shows this CIO zone containing a Non-Editable Contextual Accumulator[31] on which the user can execute simple operations like selecting or deselecting items.

[29] See chapter 5 section 2.1.3.
[30] See chapter 5 section 2.1.3.
[31] The description of this object is in appendix A section 3.3.2.2.2.

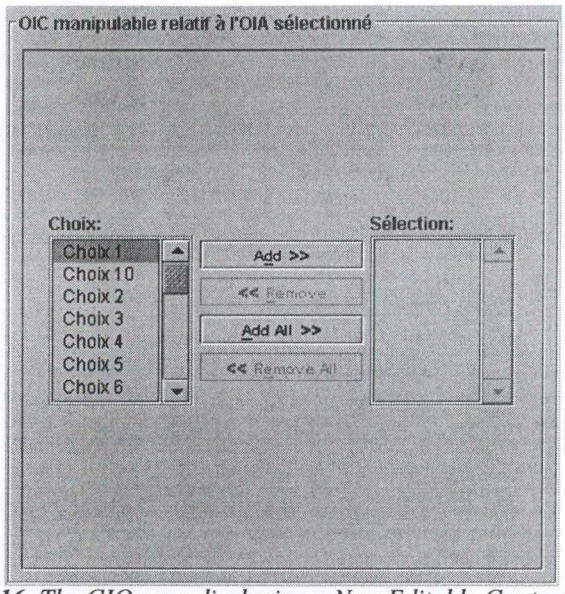


***Figure 7-16**. The CIO zone displaying a Non-Editable Contextual Accumulator*

5.1.2.3. The Attributes zone

The Attributes zone will implement the *indirect manipulation* principle[32].

This zone displays the attributes of the CIO manipulated and allows the student either to change their values or to evaluate them.

– **Attributes values**

This object is quite similar to other object inspectors that are used in some Integrated Development Environments (IDE) like Delphi or Visual Basic. In those IDE, the user can entirely customise his components (figure 7-17).

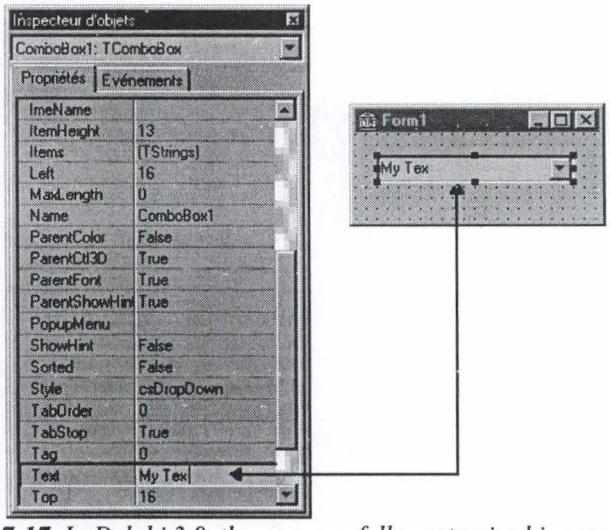


***Figure 7-17**. In Delphi 3.0, the user can fully customise his components*

[32] See chapter 5 section 2.1.3.

In this application, the objective is not to allow the student to customise the CIO, but is rather to allow him to understand what are its attributes. Therefore, it would not be very useful to let the user type any possible values for an attribute. We have decided to provide for each attribute a set of values that are typical for it. For instance, the most often used values for a "Label" attribute are strings such as "Name", "Address", and so on (figure 7-18).

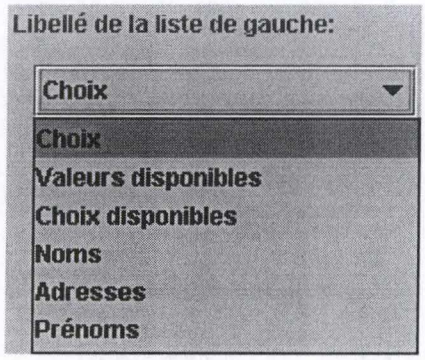


***Figure 7-18**. The possible values for the "Label" attribute of a CIO*

– **AIOs selected for the attributes manipulation**

Some attributes are quite complex and would require even more complex AIOs in order to attribute values to them. For instance, the "Selected Values" attribute of a Non-Editable Contextual Accumulator is composed of a string list. The right AIO to manipulate this attribute would then be the Non-Editable Contextual Accumulator itself. Even if this would certainly improve the *bootstrapping* of the application, we have decided not to use those complex AIOs. It would indeed be quite confusing to have to manipulate a particular component to be able to understand how the same component works. Even if this approach is pedagogically interesting and would deserve to be more thoroughly studied, we have decided to use only two basic AIOs to manipulate the attributes.

The components used are the following:

- A Drop-Down List Box for the attributes that are not composed of multiple values (figure 7-18). It is indeed one of the most basic components available and therefore, we assume that the student is at least able to manipulate it. It is almost the only prerequisite needed to use this application.
- A Multi-Line Label[33] (in a scroll pane) for the attributes that are composed of multiple values (figure 7-19). This means that the student can only modify this attribute by manipulating directly the CIO.

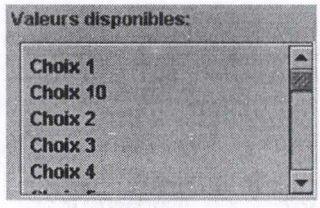


***Figure 7-19**. The "Possible values" of a CIO are displayed in a Multi-Line Label*

[33] The description of this object is in appendix A section 3.3.2.2.2.

- **Attributes definitions**

In order to implement the *concepts manipulated definitions* principle[34], the attributes will be defined as hints that appears when the user rolls the mouse over their names (figure 7-20).

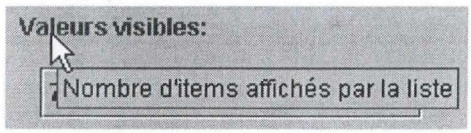


***Figure 7-20**. The attributes are defined as hints.*

- **Attributes values assignments**

In order to implement the *double reading* principle[35], the changes made on an attribute are reflected immediately on the manipulated CIO while the changes made directly on the manipulated CIO are reflected immediately on the related attributes.

The figure 7-21 shows how the assignment of the value "14" to the "Visible Rows" attribute of the Non-Editable Contextual Accumulator is reflected on the manipulated CIO.

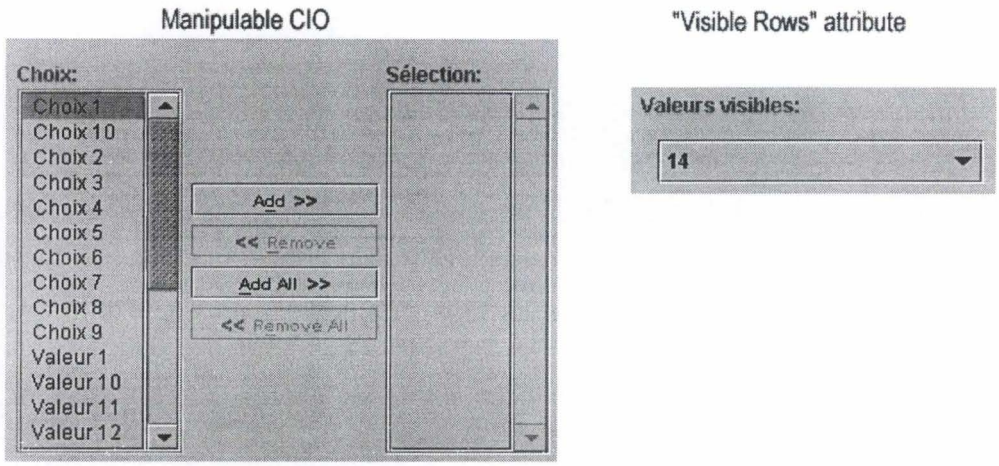


***Figure 7-21**. The assignment of the value "14" to the "Visible Rows" attribute of the Non-Editable Contextual Accumulator is reflected on the manipulated CIO.*

The figure 7-22 shows how the selection of items in the manipulated CIO is reflected in the "Selected Values" attribute.

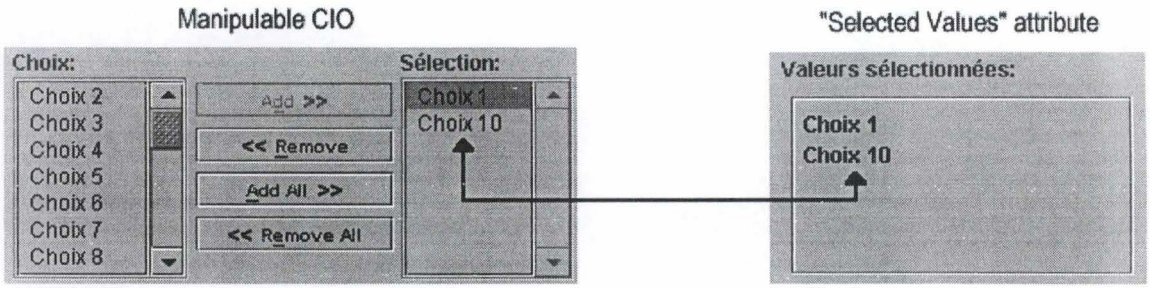


***Figure 7-22**. The selection of items in the manipulated CIO is reflected in the "Selected Values" attribute.*

[34] See chapter 5 section 2.1.3.
[35] See chapter 5 section 2.1.3.

- **Attributes list**

The attributes of the manipulated CIO are displayed in a scroll pane (figure 7-23).

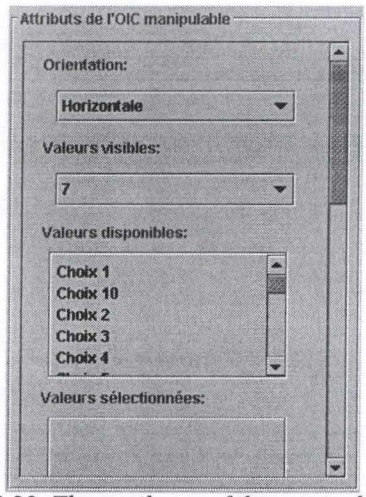


***Figure 7-23**. The attributes of the manipulated CIO*

5.1.2.4. The Application Interface

The figure 7-24 shows the application interface.

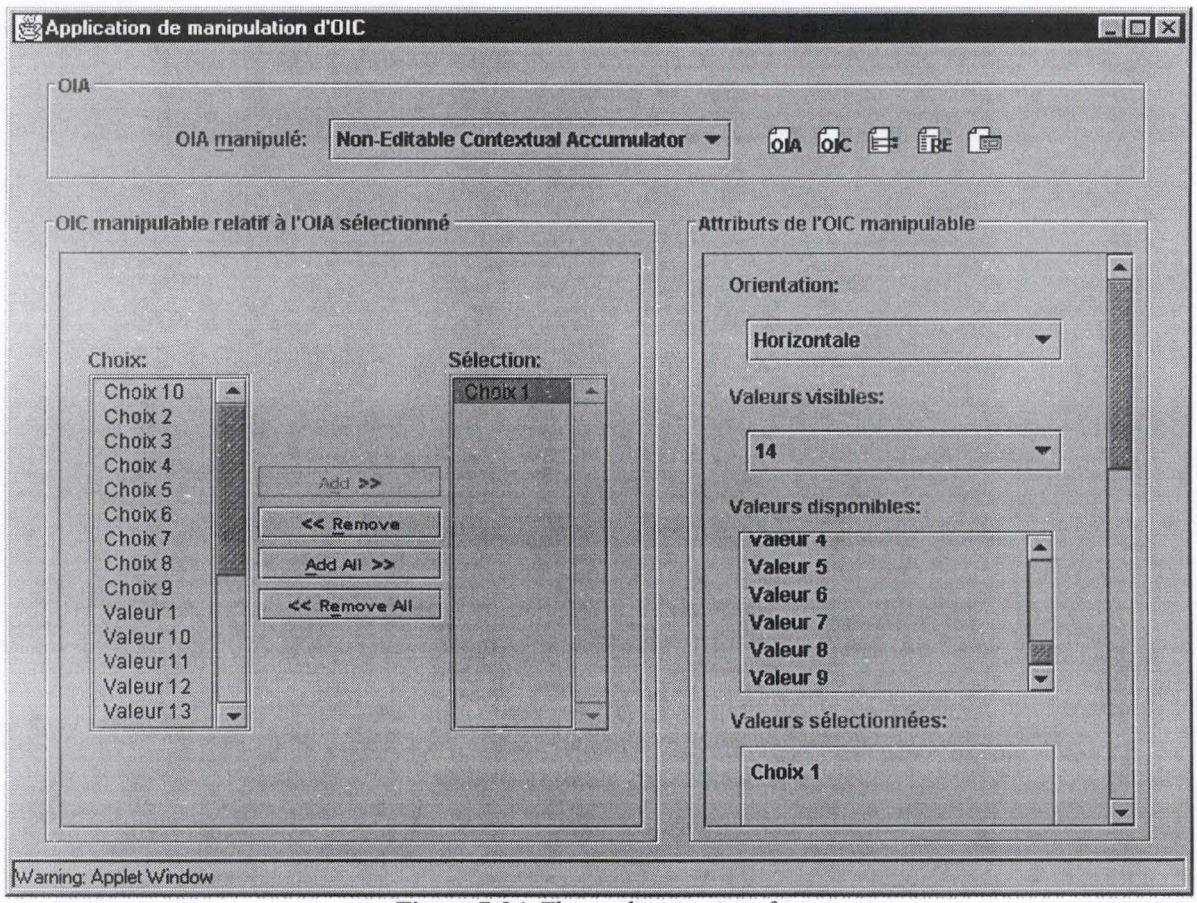


***Figure 7-24**. The application interface*

5.1.3. Application architecture

The main challenge in designing this application was to be able to reuse the High-Level Components that we had developed during our stay at the University of Port-Elizabeth without having to modify them.

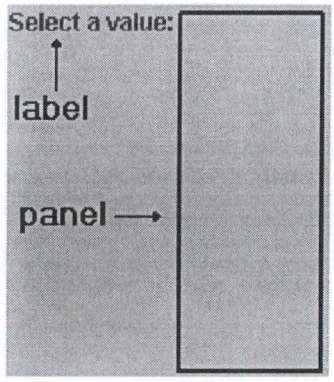


***Figure 7-25**: A labelled component with the label on the left of the panel.*

As explained in appendix A[36], most of those components extend the *LabelledComponent* class. The labelled component is the combination of a label with a panel in which it is possible to add components (figure 7-25).

In terms of relationships, we can say that the *LabelledComponent* class is inherited by from 0 to N components, while the components inherits from one and only one *LabelledComponent* (figure 7-26).

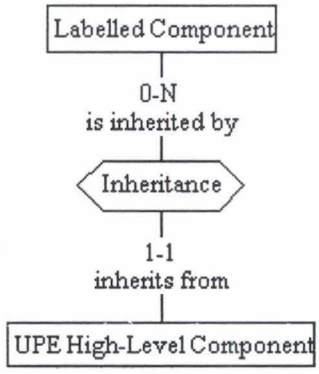


***Figure 7-26**. The ERA diagram of the relationships between the components and the Labelled Component class*

To be able to plug in those components in our applications, we had to make the following changes:

- The Labelled Component class doesn't extend the *JPanel* class anymore, but now extends the *AbstractCIO* class. This *AbstractCIO*[37] class contains a set of empty standards methods and attributes covering all the methods and attributes that could be used in the library of components.

[36] See appendix A section 4.2.2.2.1.
[37] The source code for this class is in appendix F section 1.1.

- The High-Level components are inherited by one and only one subclass which adds to this component the list of its attributes[38].

The figure7-27 illustrates the relationships between those classes.

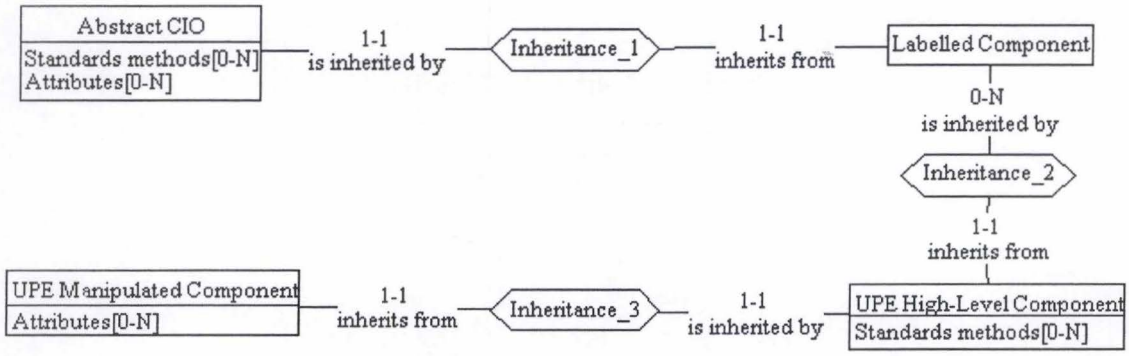


Figure 7-27. *Relationships between the main classes*

From all this, we can conclude by saying that the CIO zone can add an instance of any component to the panel, and then call any methods defined in the *AbstractCIO* class. The application knows exactly what are the methods that are overridden by the UPE High-Level component by looking at the attributes list overridden by its subclass. As our components complies with the Java Beans conventions[39] this matching is trivial.

5.2. AIOs selection trees manipulation application

In chapter 5[40], we have described the features that this application should provide.

5.2.1. Application description

We can isolate four zones that each implement specific features:

- The criteria zone, which allows the user to select selection criteria
- The AIOs partition zone, which presents the AIOs as two distinct sets according to the fact that they can be selected or not
- The Advice zone, which gives instructions to the user about what he can do
- The AIO zone, which displays information and links about AIOs

5.2.1.1. The Criteria zone

This zone will implement both the *indirect manipulation* and the *double reading* feature[41].

[38] The source code for an example of subclass is in appendix F section 1.2.
[39] The definition of Java Beans is provided in appendix A section 4.2.1.2.
[40] See chapter 5 section 2.4.
[41] See chapter 5 section 2.2.4.

– **Tree selection**

The first choice that the user must make is the specific selection tree[42] in which he wishes to navigate. Therefore, the first criteria that the he must select is the interaction type. If the interaction type selected is the input/output one, he must then select the type of information handled.

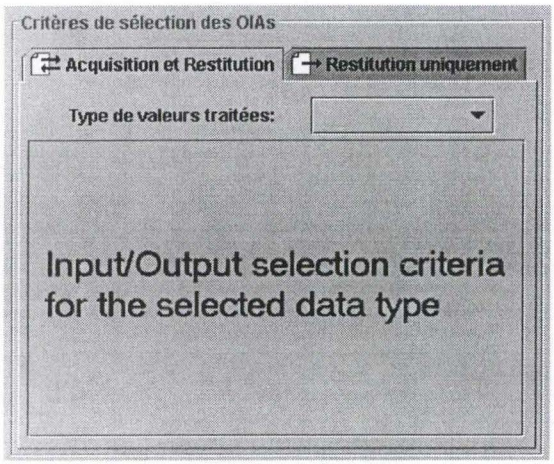


***Figure 7-28**. The Input/Output panel*

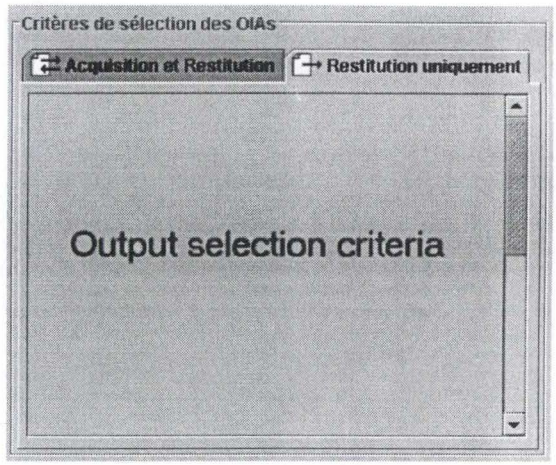


***Figure 7-29**. The Output panel*

The AIOs selected for the selection of the interaction type is a Tabbed Pane, while the AIO selected for the value type is a Drop-down List Box. The figure 7-28 shows the input/output (Acquisition/Restitution) panel, while the figure 7-29 shows the Output (Restitution uniquement) panel.

– **Criterion value assignment**

In order to assign a value to a selection criterion, we decided to use a Drop-Down List-Box displaying the possible values for each criterion(figure 7-30).

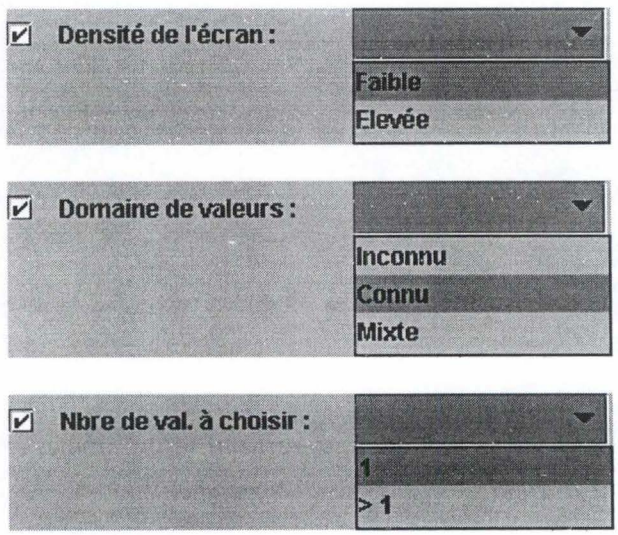


***Figure 7-30**. Selection criteria*

[42] See chapter 5 section 2.2.2.

A student must not compulsorily assign a value to a specific criterion. Therefore, each Drop-Down List Box will be preceded with a Check Box[43] allowing him either to assign a value or to remove a value previously assigned for this criterion by disabling the Check Box. The figure 7-31 shows a criterion in its two states.

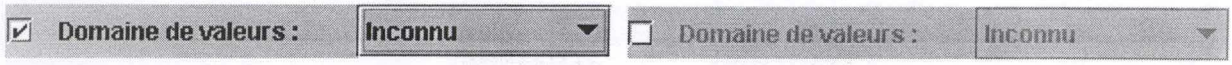


Figure 7-31. A selection criterion in its two states (enabled on the left and disabled on the right)

– **Criterion definition**

We have also implemented the *concepts manipulated definition* principle[44] in this application. The labels naming the criteria are in fact links to the definition page of those criteria on the on-line syllabus (figure 7-32).

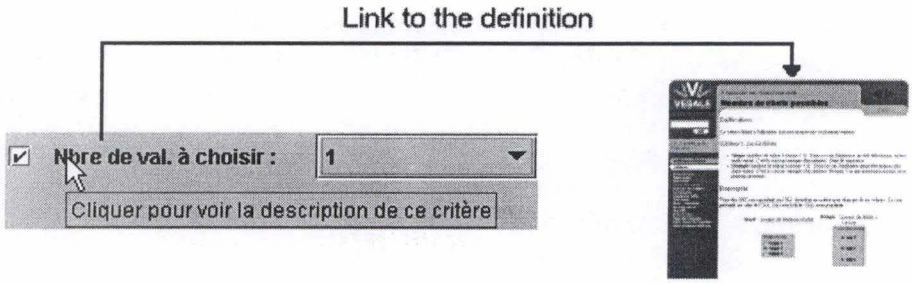


Figure 7-32. The label identifying a criterion links to the definition page of the criterion

– **Double reading**

In order to implement the *double reading* feature[45], a message will be displayed below each criterion describing the implications in term of expected features for the objects that can be selected. The figure 7-33 illustrates the double reading message if the value of the "Density" criterion is "Low" while the figure 7-34 illustrates this message if the value is "High".

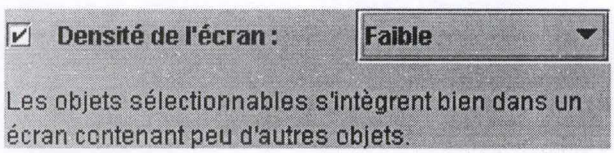


Figure 7-33. If the value assigned to this density criterion is "Low", then the message states that the objects that can be selected must fit well in a screen that contains few other objects

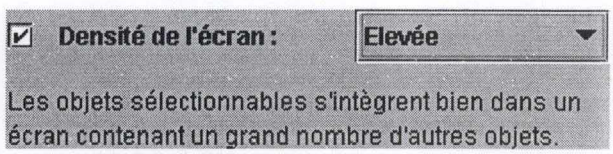


Figure 7-34. If the value assigned to this density criterion is "High", then the message states that the objects that can be selected must fit well in a screen that contains a lot of other objects

[43] The description of this object is in appendix A section 3.3.2.2.2.
[44] See chapter 5 section 2.2.4.
[45] See chapter 5 section 2.2.4.

– **Criteria list**

The list of criteria used in the selected sub-tree is presented in a panel that displays a scroll bar if the number of criteria is more than two (figure 7-35 and 7-36).

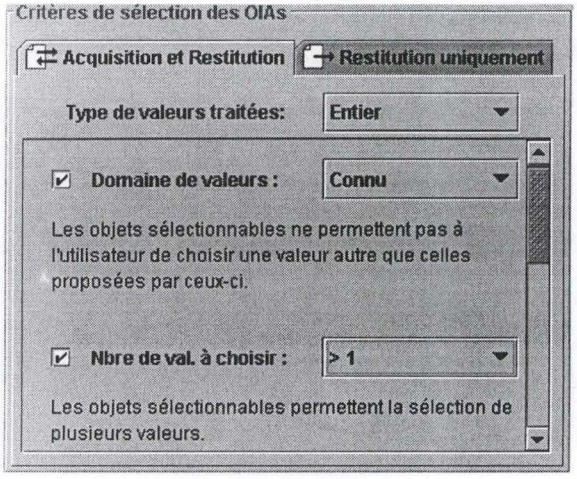


***Figure 7-35**. The criteria list for the Input/Output – Integer sub-tree*

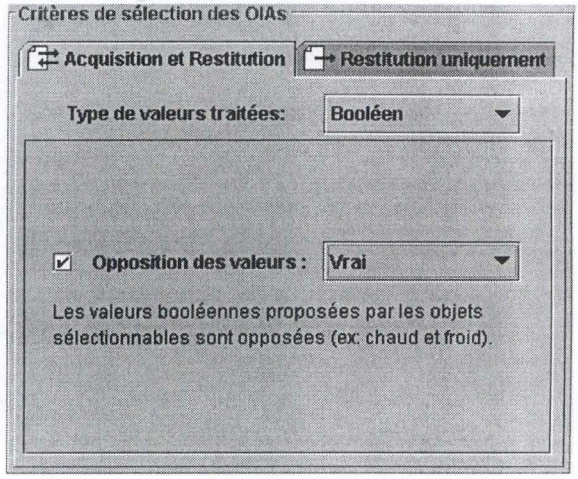


***Figure 7-36**. The criteria list for the Input/Output – Boolean sub-tree*

5.2.1.2. The Red/Green zone

This zone will present the set of AIOs included in the selection tree currently displayed. The objects that we used are List Boxes[46]. The background colour of the list that displays the AIOs that can be selected is green, while the colour of the second list is red. The same colours will also be used in the advice zone.

– **Initialisation**

When the user selects a selection tree, the green list is initialised. It contains all the AIOs that are part of the selected tree (figure 7-37).

[46] The description of this object is in appendix A section 3.3.2.2.2.

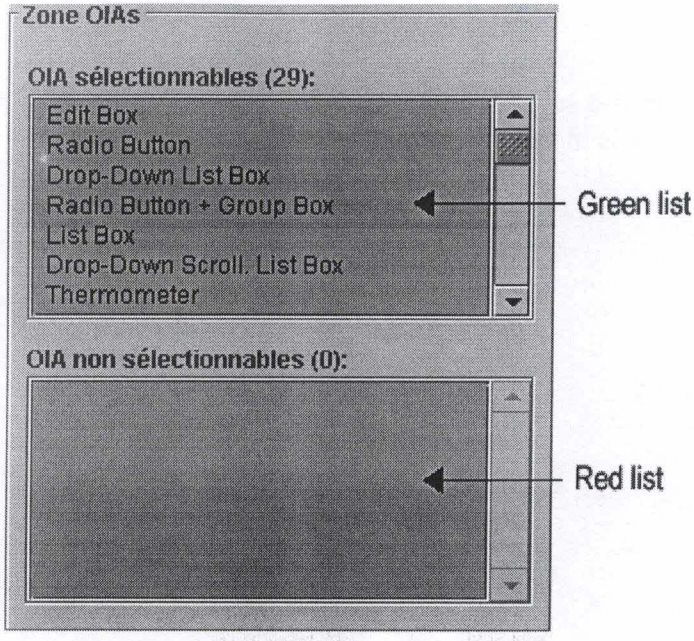


Figure 7-37. A the top, the green list contains all the AIOs contained in the selected sub-tree

- **Criterion change**

When the user assigns a value to a specific criterion, this zone must be updated.

If the user assigns a value to some criteria, part of the AIOs displayed in the green list can not be selected according to the value assigned to those criteria. Consequently, those objects are moved to the red list (figure 7-38).

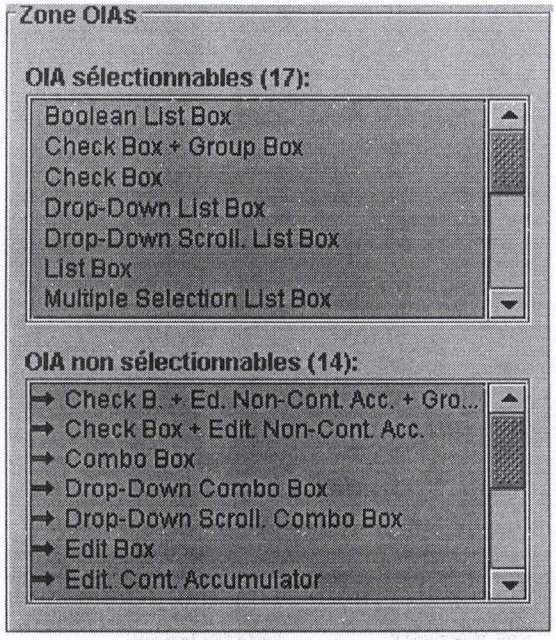


Figure 7-38. If the user assigns a value to a criterion, 14 AIOs can't be selected according to the value of this criterion.

If the user changes the value assigned to a criterion, some AIOs displayed in the green list moves to the red list and inversely. To indicate that an object has switched from lists, it is displayed at the top of the list and is preceded by an arrow (figure 7-39).

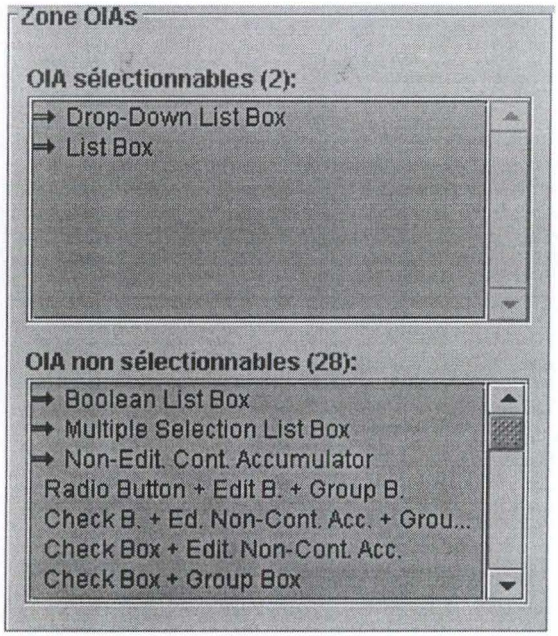


***Figure 7-39**. If the user changes the value assigned to a criterion, two AIOs that could not be selected are now in the green list while three other AIOs moved from the green list to the red list.*

5.2.1.3. The Advice zone

This zone will implement the *instruction* principle[47]. It will provide instructions about what the user can do and gives him the opportunity to execute them automatically.

The figure 7-40 displays the advice zone after its initialisation.

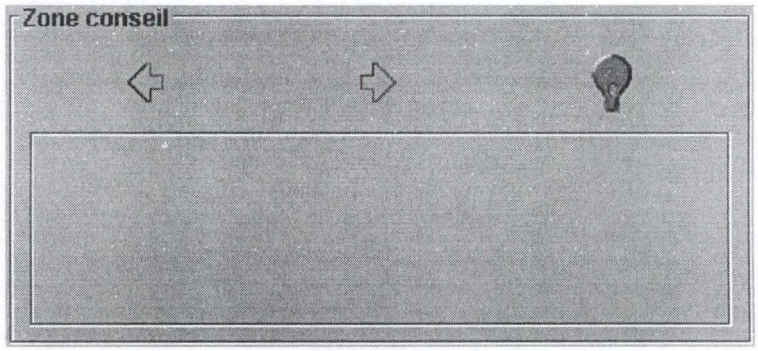


***Figure 7-40**. The advice zone after its initialisation*

[47] See chapter 5 section 2.2.4.

- **Selection of an AIO in the green list**

When the user selects an AIO in the green list, a green advice is displayed in the advice zone.

A *green advice* indicates to the user the criteria assignments that he should make in order to have only the selected AIO displayed in the green list. The light bulb button at the top of the zone is green to indicate that the advice displayed is a green one (figure 7-41).

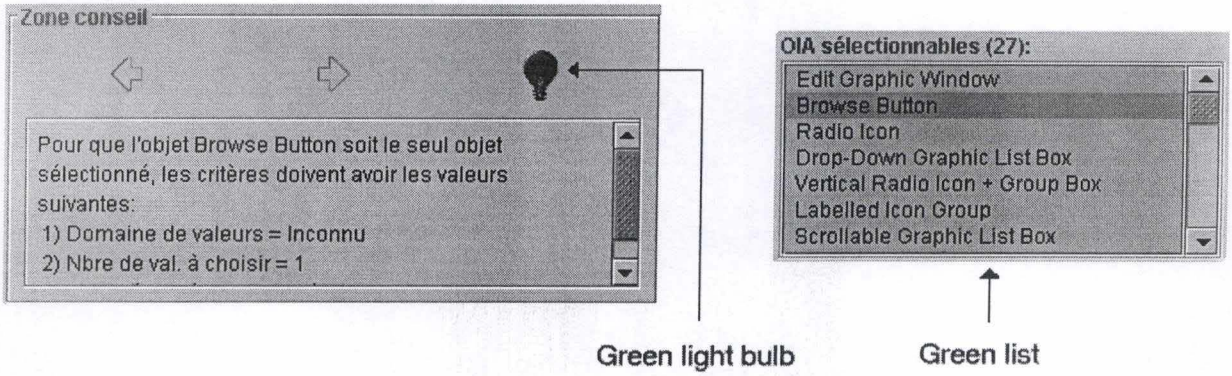


***Figure 7-41**. A green advice*

If the user clicks on the green light bulb button, the advice is then executed automatically, and the selected AIO is the only object in the green list (figure 7-42).

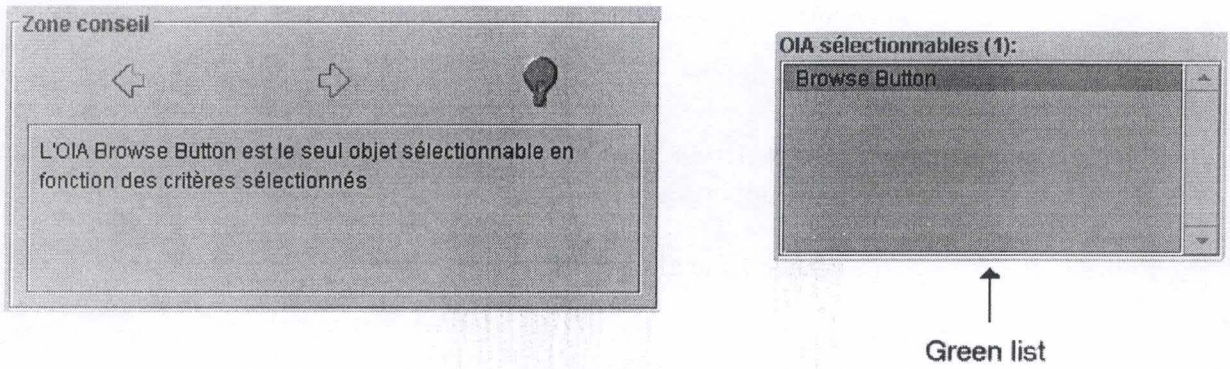


***Figure 7-42**. The green advice of the figure 7-41 has been executed automatically.*

- **Selection of an AIO in the red list**

When the user selects an AIO in the red list, a red advice is displayed in the advice zone.

A *red advice* indicates to the user the criteria assignments that he should make in order to have the AIO displayed in the green list. The light bulb button at the top of the zone is red to indicate that the advice displayed is a red one (figure 7-43)

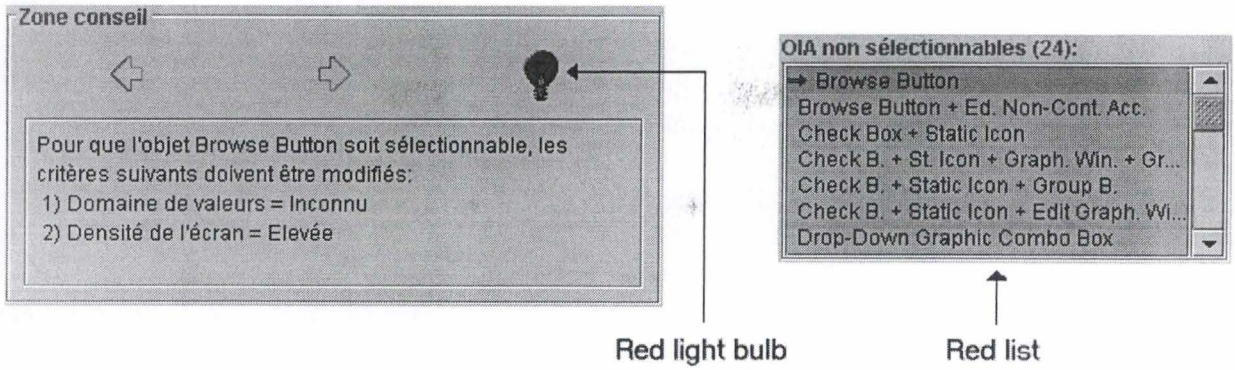


***Figure 7-43**. A red advice*

If the user clicks on the red light bulb button, the advice is executed automatically, and the selected AIO is moved to the green list (figure 7-44). At that time, a green advice is displayed and the user can execute it to make the selected AIO the only one that can be selected.

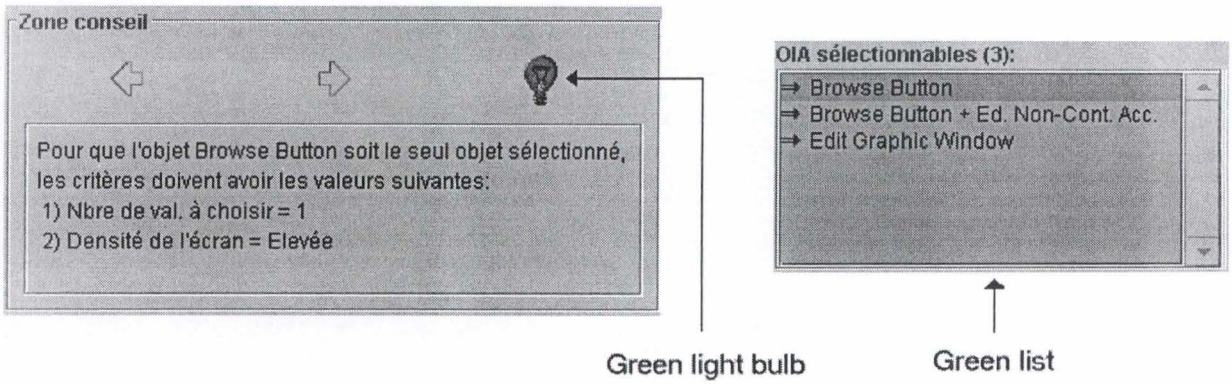


***Figure 7-44**. The red advice of the figure 7-43 has been executed automatically.*

– **Multiple advises**

A green or red advice can be composed of several assignment lists. In that case, the user has the opportunity to move from one advice to another using the arrow buttons at the top of the zone (figure 7-45).

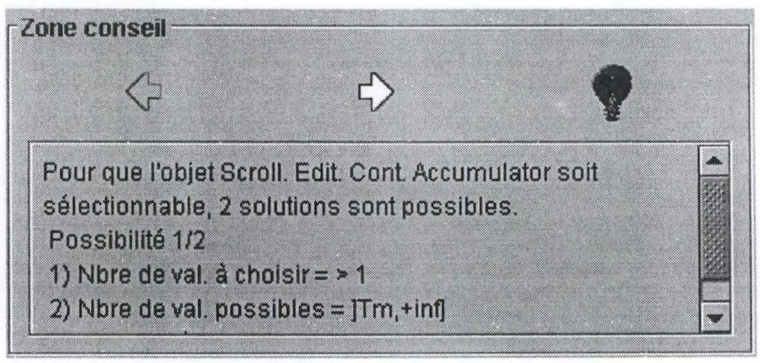


***Figure 7-45**. The arrow buttons allows the user to move from one advice to another.*

5.2.1.4. The AIO zone

The AIO zone implements the *related links* and *concepts manipulated definition* principles[48].

When the user selects an AIO in either the red or green list, the AIO zone is updated and displays information and links about this object (figure 7-46). The links are identical to those provided in the manipulation application (5.1.2.1.).

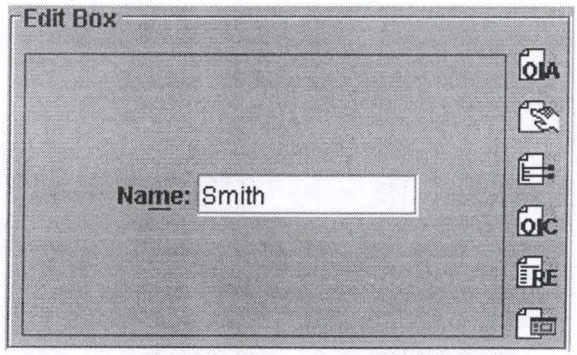


***Figure 7-46**. The AIO Zone displays information about the selected AIO*

5.2.1.5. The application interface

The figure 7-47 shows the application interface.

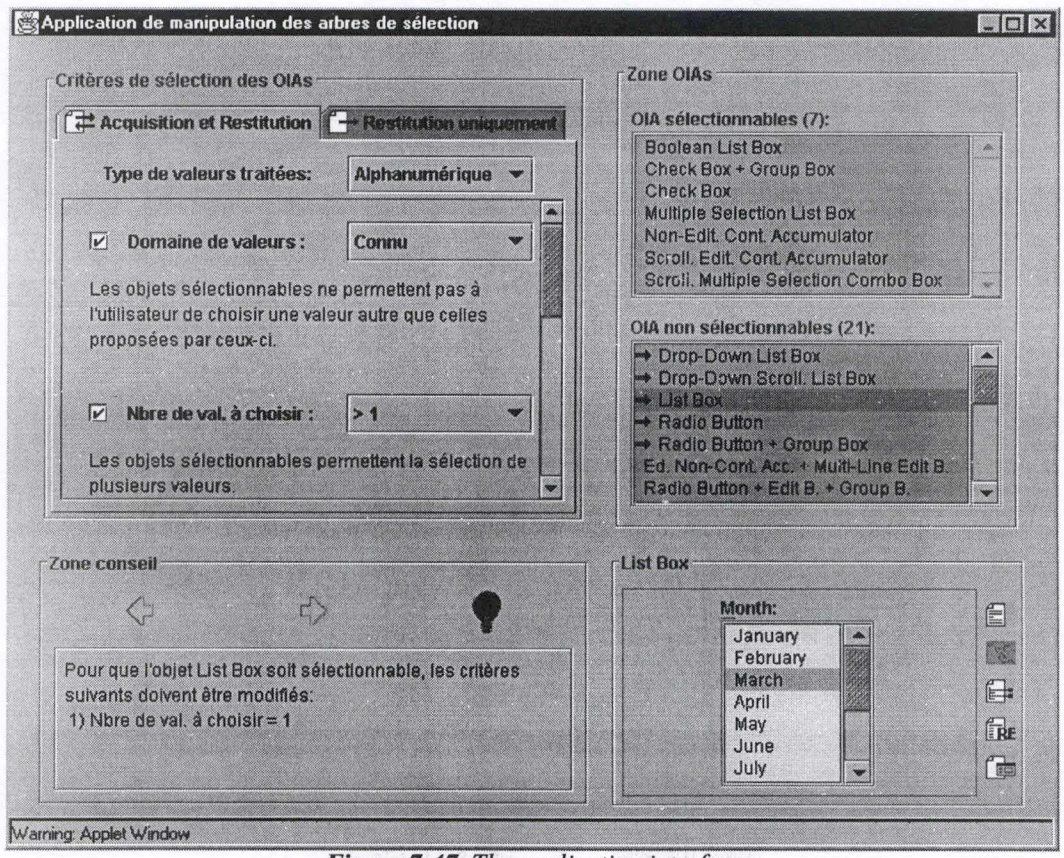


***Figure 7-47**. The application interface*

[48] See chapter 5 section 2.2.4.

5.2.2. Application architecture

The architecture of this application is composed mainly of six modules (or objects as Java is an Objet-Oriented language) :

- **The AIO zone module**

The goal of the *AIO zone* module is to display information about AIOs. This object implements the AIO zone (5.2.1.4.).

- **The Advice zone module**

The goal of the *Advice zone* module is to display advises. This object implements the Advice zone (5.2.1.3.).

- **The Red / Green zone module**

The goal of the *Red / Green zone* module, is to display the partition of the AIOs. This object implements the Red / Green zone (5.2.1.2.).

- **The Criteria zone module**

The goal of the *Criteria zone* module is to allow the user to assign values to selection criteria. This object implements the Criteria zone (5.2.1.1.).

- **The Coordinator module**

The goal of the *Coordinator* module is to listen to the events generated by the other modules, and to dispatch them by calling methods of other modules.

- **The SelectionTree module**

The goal of the *SelectionTree*[49] module is to send to the other modules information about the selection trees and the AIOs handled.

The information concerning the selection tree is taken from a text file describing its structure[50] (figure 7-48). This allows the teacher to modify the selection tree without having to modify the application code.

The information concerning the AIOs (such as its english name, its representation, ...) are taken directly from the IOs database (3.). However, as the IOs database is not yet filled with content, a text file contains the information concerning the AIOs (figure 7-48). A few methods should be rewritten in this *SelectionTree* object in order to extract information directly from the database.

[49] The code of the SelectionTree Java class is in the appendix F section 2.3.
[50] The syntax for this text file as well as the selection tree that we used are in the appendix F sections 2.1. and 2.2.

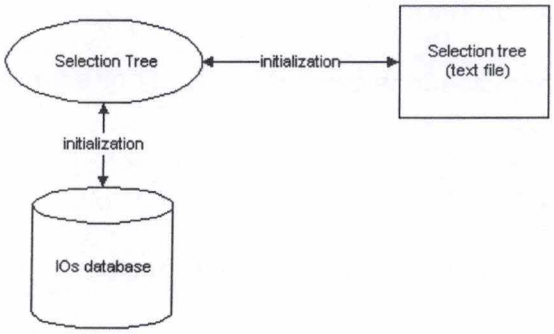


***Figure 7-48**. The SelectionTree object*

This *SelectionTree* contains all the "intelligence" of the application. When a module needs information about either AIOs or the selection tree, it communicates with this object (figure 7-49).

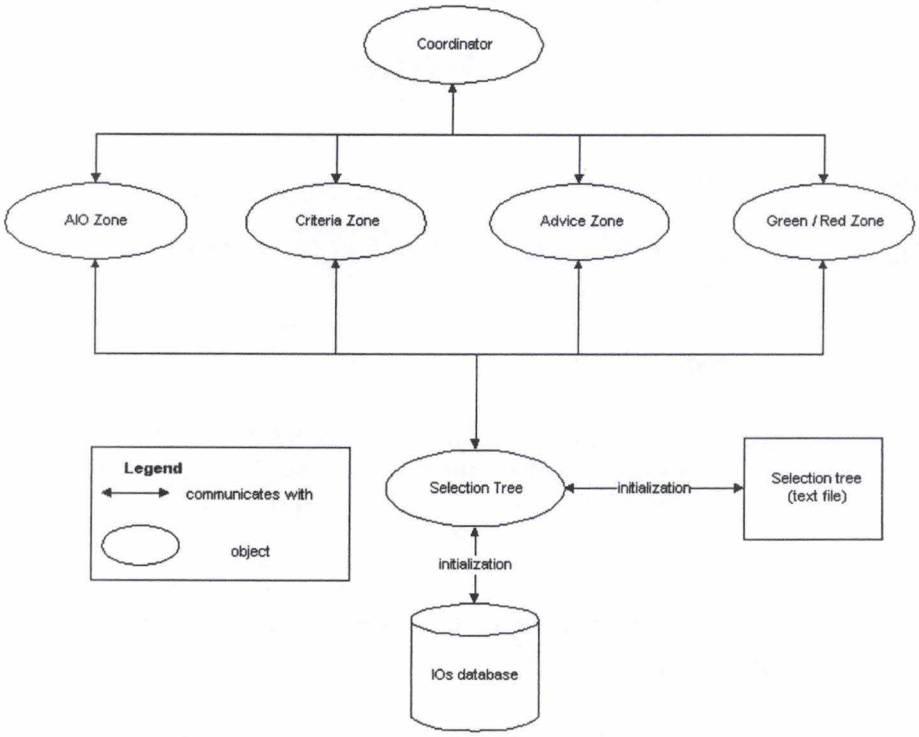


***Figure 7-49**. The application main modules architecture*

6. Course design

To design the course notes, we had to map each information chunk defined in chapter 5[51] to a web page. As advised in chapter 4[52], the content has been rewritten according to the guidelines of conciseness, scannability and objectivity. Moreover, we have implemented the conceptual navigation described in chapter 5[53].

[51] See chapter 5 section 4.1.
[52] See chapter 4 section 2.3.
[53] See chapter 5 section 4.3.

6.1. Methodology

The following methods have been used during the rewriting process:

- Use of bulleted lists
- Highlighting of keywords
- Reduction of "useless" words or sentences

Moreover, when we met a concept defined in another chunk, we included a link to the web page defining it.

6.2. Example: the "AIO Definition" chunk

The following example illustrates the way we proceeded in order to map content for the "AIO definition" chunk. The text in the figure 7-50 is excerpt from the paper course.

Du concept d'OIC découle un **problème de généralisation**: un même OIC peut se retrouver dans plusieurs environnements physiques sous diverses appellations. Si seul l'outil de présentation change, seule la représentation graphique change; si, en plus, l'outil graphique varie, le comportement de l'OIC peut connaître des variantes.

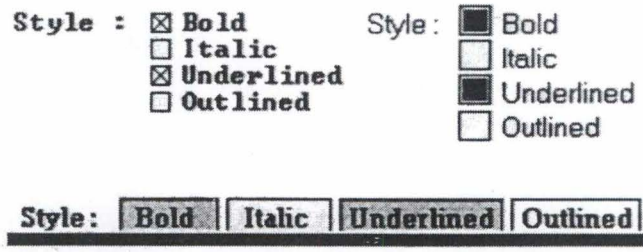


Figure IL 10: Un même objet interactif dans différents environnements

Par exemple, le même OIC de la Figure II.10 existe dans différents environments : «Check box» dans *Ms-Windows,* «XmToggleButton» dans *OSF/Motif,* «BoxArray» dans *Garnet.* Pour pallier cette carence, le concept d'objet interactif abstrait est envisagé.

Un *objet interactif abstrait* (OIA) constitue une abstraction de l'ensemble des OIC de même type indépendamment des environnements physiques qui l'accueillent. Chaque OIA est décrit selon un modèle en deux sections:

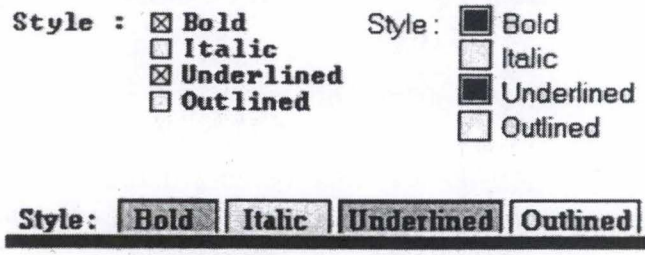
1. Une section descriptive des caractéristiques de l'OIA : outre les noms et abréviations, on y définit l'OIA, sa nature, son type et ses relations éventuelles d'agrégation;
2. Une section comportementale : on y décrit les relations d'héritage, les opérations permises (i.e. les causes de déclenchement et leurs effets), les attributs abstraits, les événements abstraits et les primitives abstraites spécifiant le comportement de l'objet. Cette description orientée objet respecte les propriétés d'**encapsulation** (la gestion de l'objet s'effectue par ses primitives abstraites seules) et d'**héritage** (les objets de tout sous-type d'un type héritent des propriétés du sur-type et peut posséder, en plus, des propriétés spécifiques).

Figure 7-50*. The AIO definition part of the paper course*

The text in figure 7-51 illustrate the result of the rewriting and linkage process.

Du concept d'OIC découle un **problème de généralisation**: un même OIC peut se retrouver dans plusieurs environnements physiques sous diverses appellations. Si seul l'outil de présentation change, seule la représentation graphique change; si, en plus, l'outil graphique varie, le comportement de l'OIC peut connaître des variantes.

Figure II-10. Un même OIC dans trois environnements différents.



Un **objet interactif abstrait** (OIA) constitue une abstraction de l'ensemble des OIC de même type indépendamment des environnements physiques qui l'accueillent. Chaque OIA est décrit selon un modèle en deux sections:

1. Les caractéristiques de l'OIA :
   - Nom
   - Abréviations
   - Définition
   - Nature
   - Type
   - Relations éventuelles d'agrégation

2. Le comportement de l'OIA :
   - Relations d'héritage
   - Opérations permises
   - Attributs abstraits
   - Evénements abstraits
   - Primitives abstraites

Cette description orientée objet respecte les propriétés

- **D'encapsulation** : la gestion de l'objet s'effectue par ses primitives abstraites seules.
- **D'héritage** : les objets de tout sous-type d'un type héritent des propriétés du sur-type et peut posséder, en plus, des propriétés spécifiques.

Figure 7-51. *The AIO definition information chunk*

We brought the following transformations to the original text:

- We reduced the words count by 25% (169 instead of 225). It is less than the 50% reduction advised by Jakob Nielsen[54]. The reason is that, basically, a course contains less useless words than a business oriented brochure.
- We substituted every enumeration with bulleted lists.
- We transformed the concepts defined in another chunk into hypertext links.
- We repositioned the legend at the top of the figure. We did it to prevent a figure to be displayed with its legend hidden due to the limited vertical view of the page on a screen.

7. Conclusion

In this chapter, we described the design process that we followed. Now that we have implemented this prototype, we have to evaluate what is its contribution to the learning of Interaction Objects. This is the purpose of the next chapter.

[54] See chapter 4 section 2.3.1.

8

System Evaluation

1. Introduction

In this chapter, we will evaluate the system as implemented in the previous chapter. As *bootstrapping* is the most important design principle of the VESALE project, we will first look into how our interactive applications have implemented it. Secondly, we will try to evaluate what such a system can bring to the learning process of the Interaction Objects (IOs).

The present reflection is based on our own personal experience of the university courses that we had to study and not on pedagogical theories.

2. Bootstrapping

At our level, *bootstrapping* means that the AIOs selected to build our applications will be the best AIOs possible for the particular context in which they are used. As we explained in the design of our applications[1], the AIOs used have been selected according to the selection trees presented in the course.

It is interesting to notice a few cases in which the *bootstrapping* principle is applied at a higher level. For instance, the figure 8-1 shows the selection tree application on which the only AIO that can be selected according to the values that the student assigned to the criteria is the Static Icon[2] object.

---

[1] See chapter 7 section 5.
[2] The description of this object is in appendix A section 3.3.2.2.2.

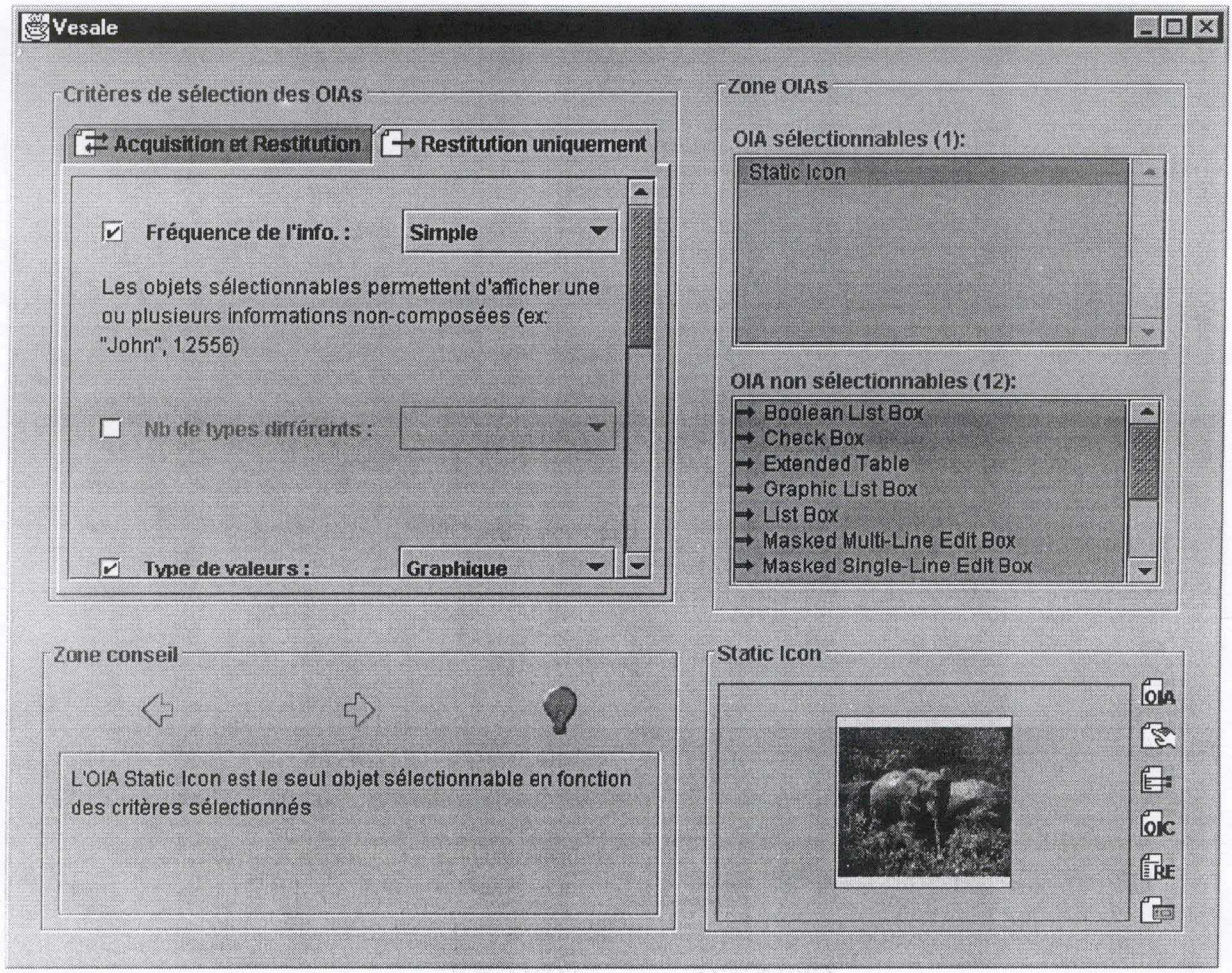


***Figure 8-1**. The selection tree application*

The AIO zone will then display a picture of this Static Icon AIO. The AIO used for this feature is a Static Icon, which means that the Static Icon picture is displayed inside the same Static Icon component. This will hopefully reinforce the feeling of the student that a Static Icon AIO must be used when only one graphical information must be given to the user.

Another example of *bootstrapping* is when the selected AIO in the IOs manipulation application is a Drop-Down List Box[3] (figure 8-2). As the AIOs used to manipulate this AIO are mostly Drop-Down List Boxes, this application is even more efficient because the student manipulates twice as much the Drop-Down List Box AIO as other manipulated AIOs.

[3] The description of this object is in appendix A section 3.3.2.2.2.

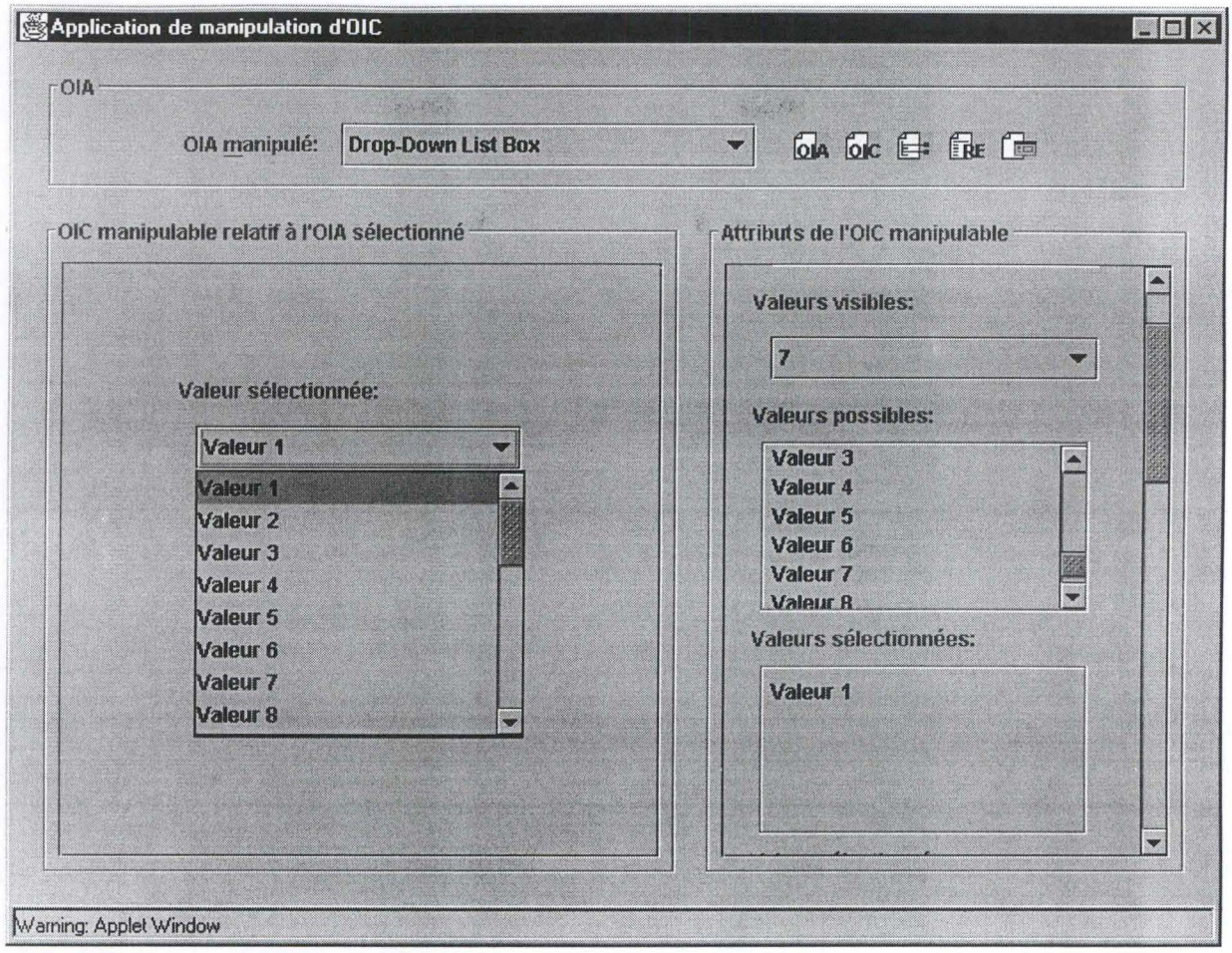


***Figure 8-2**. The manipulation application when the manipulated AIO is the Drop-Down List Box*

3. Contribution to the learning of IOs

In this section, we will evaluate our web version of the course. We will base this evaluation on our experience as students of the Institut d'informatique who had to learn a HCI course including the part concerning the IOs[4]. We will wonder if such an environment would have helped us in the learning of this course and more generally what could be the contribution of a multimedia environment to the learning.

First, we will evaluate what is the contribution of the on-line course in comparison with the paper syllabus. Secondly, we will try to estimate what is the potential pedagogical contribution of this environment.

[4] We followed this course as part of our "2ème maitrise". It was given at that time by Jean Vanderdonckt.

3.1. Support contribution

We will try to weigh what is gained and what is lost in the on-line version of the syllabus in comparison with the paper course version. This comparison concerns essentially the supports of the course, namely a multimedia computer and a paper syllabus.

In order to stay focussed on what is really important, we will not compare the web course with the actual paper course but rather with an "ideal course[5]". This means that considerations like the typographic consistence and the page structure will not be taken into account.

3.1.1. What we have gained

- **The conceptual navigation**

When the student encounters a concept that he doesn't understand in the paper course, he has to search through the whole syllabus in order to find its definition. On the opposite side, for every concepts used in the definition of another concept in the web course, a link to its definition page is provided. The figure 8-3 shows the web version of the AIO definition. The definition in itself is the same as in the paper course but in the web version the student can easily access related concepts definitions.

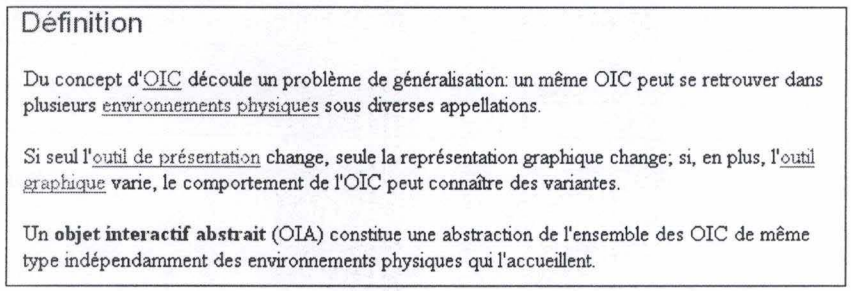


***Figure 8-3**. The AIO definition contains links to the concepts of CIO, physical environment, presentation tools and graphical tools.*

- **Hierarchical navigation**

If the student wants to know where he is in a paper course, he can always refer to the table of contents. What the web version of the course adds, is that the hierarchy **as well as** the present position in this hierarchy are always displayed. Therefore, the student always keeps in mind the course structure. The figure 8-4 shows clearly that in our web course, the hierarchy is developed for the current section and the current page is highlighted.

Moreover, this representation of the course structure allows the student to access directly a desired information without having to locate it in the middle of less important information. A typical example in this course is the description of the attributes, events and primitives of the AIOs. In the paper course, if the student is not interested in these AIOs features, he will have to turn a few pages describing them between each AIO description. In comparison, he can reach directly the desired information in the web course.

[5] We means a course with a consistent layout, a table of content, a clear structure,...

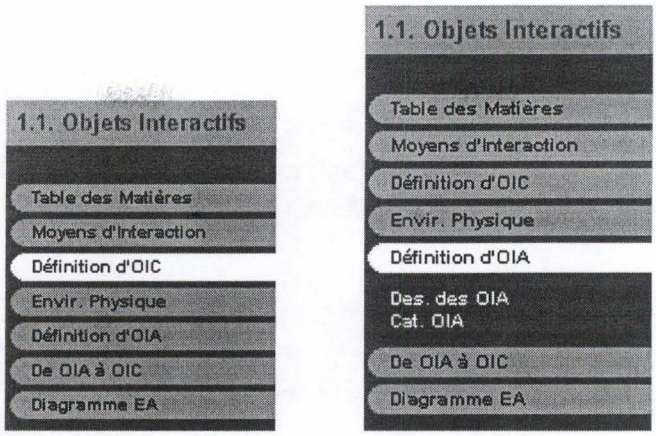


***Figure 8-4**. The firs menu shows that the user is currently reading the CIO definition. The second one tells the user that he is reading the AIO definition and that this section contains two other parts.*

- **Visual memorisation**

In a paper course, the titles structure is often page-dependent rather than structure-dependent. It means that just because there is space left on the bottom of a paper page, a section title can be separated from its content. Moreover, a paper page is not extensible which means that a concept can hold on a half page or be spread across several ones.

In our web course, each page begins with its concept title and contains only what a user can expect from that title (figure 8-5). This is not only a layout consideration but we think that it increases the possibilities of visual memorisation of the concept presented in the page. According to our experience, it is indeed easier to memorise a concept if a unique visual representation is provided.

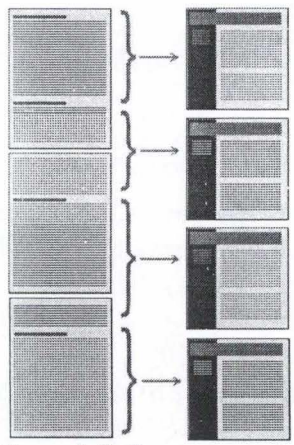


***Figure 8-5**. Content mapping.*

- **Support capabilities**

In a paper course, the illustrations are in black and white and are often photocopied copies of poor quality. Moreover, this support enables no interaction. On the opposite side, a multimedia support offers the possibility to present colourful illustrations and interactive applications.

3.1.2. What we have lost

- **Portability**

While a web based course needs a computer and a connection to the Internet or to the university intranet, a paper course can be read anywhere.

- **Support qualities**

A paper course offers a better visual comfort than a computer screen. A student can also annotate and highlight the text of the paper course while it is not (at the moment) possible with the web course. Moreover, notions like the course size according to the thickness of the syllabus are simply gone. In other words, students lose their traditional landmarks.

- **Sequential navigation**

Even if the web course provides a sequential navigation, it also provides a conceptual one that can be as much an advantage as a trap. The student can indeed get lost in the links and forget what he has seen and understood from what he has not. On the opposite side, the paper course eludes this issue by imposing the sequentiality .

3.1.3. Support contribution balance

We think that the ultimate advantage of a paper course over a web course is that it is the most natural learning support for us (students of the "old generation"). Consequently, our preferred support remains the paper course but we can certainly take advantage of such a multimedia learning environment as a *complementary* support. Nevertheless, we think that a generation that would have used multimedia environments as learning tools since their youngest age could consider such an environment as a *primary* learning support.

3.2. Pedagogical approach contribution

During our studies, we have noticed two different conceptions in the way a teacher introduces a particular subject. Some teachers start their course by flooding the students with theories without any illustrations. Others start with an illustration that we won't be able to resolve before the end of the theory presentation. Needless to say that we are in favour of the second approach. The reason is that, in that case, we know what we can expect from the course, why we will have to learn theory and what is the *context*.

This idea of reversing the logical presentation of a concept (examples prior to the theory) is the main application of *Constructivism*[6] that we have faced during our studies. In our context, the idea of Constructivism goes far beyond introducing a concept with an illustration. In the suggested Constructivist scenario[7], we give the opportunity to **manipulate** the concepts in order to introduce the theory in its *context*.

The concepts manipulation is not a predetermined process: what a student can deduct from the manipulation will certainly be different from what another will. This idea that each student

[6] See chapter 3 section 2.
[7] See chapter 5 section 4.2.1.2.

builds his **own** mental model is completely new for us. We don't know if this can have a positive influence on the learning process. It could lead to a much deeper understanding because the mastering of a model built by the student is stronger that the mastering of an instilled model. On the other hand, it could also turn out to be completely confusing for the student and therefore unusable. Since we are not pedagogues, this issue is beyond our reach.

4. Conclusion

Before starting developing this learning environment, we where not really convinced of its usefulness in the learning of interactive objects. As a matter of fact, the disadvantages of such a computer-based environment explained in this chapter make us think that it will likely be used as a *complementary* study support by the students in the short term.

Nevertheless, the original approach provided by such an environment seems to be a major contribution to the learning of Interaction Objects. As computer scientist, it is difficult for us to evaluate its **exact** contribution.

9

Conclusion

The objective of this thesis was the development of a multimedia environment for the learning of Interaction Objects (IOs). This work was part of the VESALE project of the Institut d'informatique. The challenge we faced was to take advantage of the freedom we had in order to develop the most efficient learning environment possible. This freedom was twofold.

First, we were free to implement whatever could contribute to a better learning process. Indeed, the only things that we had to develop were the IOs database and the course notes related. We had few constraints concerning the expected features of those elements. The most important things that we brought are probably the interactive applications and the pedagogical scenarios. On one hand, the interactive applications[1] enable the student to be active in his learning process. On the other hand, the pedagogical scenarios[2], and more precisely the Constructivist one, provide different ways of thinking the course structure.

Secondly, besides the fact that this project had to be included in a web environment, we were free to use whatever technologies could help us in order to implement it. Indeed, the technologies used for the VESALE project development were not yet chosen at the time we wrote this thesis. Therefore, this challenge was to select and master technologies whose potentials and working were unknown to us. The most important technology we have learned was certainly the Java programming language. We have received the opportunity to familiarise ourselves with it during our stay at the University of Port-Elizabeth, and we went deeper into its knowledge for this project. It allowed us to develop the interactive applications, the dynamic pages generation as well as the dynamic navigation.

---

[1] See chapter 5 section 2.
[2] See chapter 5 section 4

Finally, let's remind that we based ourselves on our own experience as computer scientist students to develop this environment. Therefore, it must be regarded as a prototype that will have to be criticised by pedagogues in order to validate (or invalidate) its pedagogical contribution.

From a personal point of view, we can say that this final year of our studies brought us a lot. Our stay in South Africa gave us indeed the opportunity to live in contact with a very friendly and humanly enriching culture. Moreover, this thesis gave us the opportunity to work on a concrete project. In such a motivating context, we have learned a lot and, therefore, prepared ourselves for our first step in the professional live.

References

[Ameritech99]
Ameritech Web Site, Ameritech Graphical User Interface Standards and Design Guidelines, *http:\\www.ameritech.com/*, 1999

[Apple99]
Apple Web Site, MacOs Graphical User Interface Standards and Design Guidelines, *http://www.apple.com*, 1999

[Badot-Detez98]
P. Badot, V. Detez, "*Vers un corpus de règles ergonomiques pour la création de sites Web*", master thesis, Institut d'informatique, FUNDP, Namur, 1998

[Beirekdar99]
A. Beirekdar, "VESALE (Visual user interface design Education Supported by a computer-Aided Learning Environment) : Specification document", *http://www.info.fundp.ac.be/~vesale/*, July 99

[Bodart93]
F. Bodart, Y. Pigneur, *Conception assistée des systèmes d'information: Méthode, modèles, outils* , Masson, 1993

[Bodart99]
F. Bodart, J-M. Leheureux, E. Mbaki, A. Beirekdar, "Definition of the project VESALE :Visual user interface design Education Supported by a computer-Aided Learning Environment", *http://www.info.fundp.ac.be/~vesale/*, July 99

[Bodart-Magnier99]
Th. Bodart, M.L. Magnier ,"The Representation of Abstract Interaction Objects for Reusable Object Design", UPE.

[Bos99]
B. Bos, "Web Style Sheets", *http://www.w3.org/Style/*, August 99

[Campbell99]
K. Campbell, "The Web: Design for Active Learning", *http://www.atl.ualberta.ca*

[Catizzone-Remy99]
M. Catizzone, E. Remy, "Virtual University", master thesis, Institut d'informatique, FUNDP, Namur, 1999

[Eckel98]
B. Eckel, *Thinking in Java*, Prentice Hall, Upper Saddle River, NJ, 1998

[Farley98]
J. Farley, *Java Distributed Programming*, O'Reilly & Associates, Inc., Sebastopol, 1998

[Fowler98]
S. Fowler, *GUI design handbook*, McGraw-Hill, 1998

[Friesen99]
G. Friesen, "Plug into Java with Java Plug-in", *http://www.javaworld.com/*, June 99

[Goodman98]
D. Goodman, *Dynamic HTML, The Definitive Reference*, O'Reilly & Associates, Inc., Sebastopol, 1998

[Hofstetter99]
Fred T. Hofstetter, "Cognitive Versus Behavioral Psychology",
http://www.udel.edu/fth/pbs/webmodel.htm

[Hunter98]
J. Hunter, W. Crawford, *Java Servlet Programming*, O'Reilly & Associates, Inc., Sebastopol, 1998

[Koyanagi99]
Mark Koyanagi, "Putting courses on-line: Theory and Practice", lost reference

[Kreines99]
D. C. Kreines, B. Laskey, *Oracle database administration*, O'Reilly & Associates, Inc., Sebastopol, 1999

[Leclercq98]
D. Leclercq, B. Denis, "Objectifs et paradigmes d'enseignements/apprentissage", *Pour une Pédagogie Universitaire de Qualité*, p 83-105, Nardaga, Sprimont (Belgium), 1998

[Lynch99]
P. J. Lynch, S. Horton, *Web Style Guide: Basic Design Principles for Creating Web Sites*, Yale University Press, New Haven and London, 1999

[Mayes90]
Mayes, "Conceptual Space Navigation"
http://cbl.leeds.ac.uk/nikos/tmp/hypemedia/subsection2_5_2_11.html

[Michiels-Prévot99]
R. Michiels, G. Prévot, "A case base as a learning tool for the design of human computer interfaces", master thesis, Institut d'informatique, FUNDP, Namur, 1998

[Morkes97]
J. Morkes, J. Nielsen, "Concise, SCANNABLE, and Objective: How to Write for the Web", *http://www.useit.com*, 1997

[Morkes98]
J. Morkes, J. Nielsen, "Applying Writing Guidelines to Web Pages ", *http://www.useit.com*, January 6, 1998

[Nielsen96a]
J. Nielsen, "Why frames suck (most of the time)", *http://www.useit.com*, December 1996

[Nielsen96b]
J. Nielsen, "In Defence of Print ", *http://www.useit.com*, February 1996

[Nielsen96c]
J. Nielsen, "The rise of the sub-site", *http://www.useit.com*, September 1996

[Nielsen97]
J. Nielsen, " How Users Read on the Web ", *http://www.useit.com*, October 1, 1997

[Nielsen98a]
J. Nielsen, "What is Usability?", *http://www.zdnet.com/devhead/stories/articles*, September 14, 1998

[Nielsen98b]
J. Nielsen, "Web Usability: why and How", *http://www.zdnet.com/devhead/stories/articles*, September 14, 1998

[Nielsen99a]
J. Nielsen, "How to Structure Your Web Site", *http://www.zdnet.com/devhead/stories/articles*, May 4, 1999

[Nielsen99b]
J. Nielsen, PJ Schemenaur, Jonathan Fox, "Writing for the Web", *http://www.sun.com*, July 1999

[PVWG98]
PVWG, *http://www.info.fundp.ac.be/~vesale/PVWG*

[Raggett99]
D. Raggett, I. Jacobs, "HyperText Markup Language", *http://www.w3c.org/MarkUp/*, February 99

[Reese97]
George Reese, Database *programming with JDBC and Java*, O'Reilly & Associates, Inc., Sebastopol, 1997

[Rosenfeld98]
Louis Rosenfeld, Peter Morville, *Information Architecture for the World Wide Web*, O'Reilly & Associates, Inc., Sebastopol, 1998

[Strommen92]
Erik F.Strommen, "Constructivism, Technology, and the Future of Classroom Learning", *http://www.ilt.columbia.edu/k12/livetext/docs/construct.html*,

[Sun99a]
Sun Microsystems, "The Java Tutorial, A practical guide for programmers", *http://java.sun.com/docs/books/tutorial/index.html*, July 99

[Sun99b]
Java Look and Feel Design Guidelines ,Alpha draft, *http://www.java.sun.com/,* June 1999

[Sun]
Sun Web site, *http://www.sun.com*

[Vanderdonckt96]
J. Vanderdonckt, *Une description orientée objet des objets interactifs abstraits utilisés dans les interfaces homme-machine (course notes)*, Institut d'informatique - Facultés Universitaires Notre-Dame de la Paix, Belgium, Namur, 1996

[Vanderdonckt97]
J. Vanderdonckt, *Conception assistée de la presentation d'une interface homme-machine ergonomique pour une application de gestion hautement interactive.* PHD Thesis . Insitut d'Informatique, Namur, Belgium, July 9, 1997

[Vanderdonckt98]
J. Vanderdonckt, J-M. Leheureux, "Vesale : un environnement de support multimédia à l'enseignement des interfaces homme-machine", *http://www-ihm.lri.fr/ihm98/contributions/20_Vanderdonkt/VesPos.htm, 1998*

[Weinschenk97]
S. Weinschenk, P. Jamar, Sarah C. Yeo, *GUI design essentials*, Wiley Computer Publishing, 1997

[Yale99a]
Yale Style Manual, *http://info.med.yale.edu/caim/manual*

[Yale99b]
Cranial Nerves, *http*

Appendixes

Table of contents

Appendix A: The Representation of Abstract Interaction Objects for Reusable Object Design

Appendix F: Interactive applications

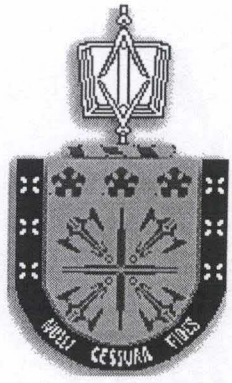
A

The Representation of Abstract Interaction Objects for Reusable Object Design

From September to January, we worked at the University of Port Elizabeth (UPE) in South Africa. Our supervisor was Professor Janet L. Wesson.

During this internship period, we developed a library of IOs in Java. The title of the report we had to write in that occasion is "*The Representation of Abstract Interaction Objects for Reusable Object Design*".

University of Port Elizabeth
Department of Computer Science and Information Systems
Port Elizabeth 6000



The Representation of Abstract Interaction Objects for Reusable Object Design

A treatise submitted by
Thibaut Bodart and Marc-Laurent Magnier

Supervisor: Prof. Janet Wesson
Co-supervisor: Mr. Leon Nicholls

Academic year 1999

1. Introduction

1.1. Foreword

This report is the result of four months of work at the Department of Computer Science and Information Systems of the University of Port Elizabeth. The project we've been working on was part of the *"A Visual Environment for Reusable Object Design" (VEROD)* project. Our supervisors were the professor Janet Wesson and Leon Nicholls.

1.2. AIOs and CIOs

The *concrete interactive objects* (CIOs) are the graphical objects for the input and the display of data that the user can see, feel and manipulate. CIO is synonymous to a control, a physical interactor, a widget or a presentation object. They have one and only one graphic representation. They are used at the *physical* level.

The *Abstract Interactive Objects* (AIOs) abstract both presentation and behaviour of CIO's. When working with AIO's, the focus is set on the behaviour of the object instead of on his graphic representation. They are used at the *logical* level.

For instance, the List Box AIO (see Figure 3.9) is an abstraction of the Macintosh CIO Scrolling Box and of the MS-Windows CIO List Box. Both CIO have the same properties, events and methods but have a different graphic representation.

1.3. Description of the VEROD project

A Visual Environment for Reusable Object Design (Nicholls, 1998)

Graphical user interfaces (GUI's) have become accepted interface standards for interactive design. More recently, object based GUI's such as Windows 95, have extended the interactivity and usability of these interfaces by encapsulating the functionality of interactive elements.

These developments impose serious burdens on programmers. Programming in these GUI's is complex and requires expertise in a wide range of fields from low-level coding techniques to Human Computer Interaction (HCI) principles required for good interface design. In addition, popular programming languages have merely been extended to support the development of GUI's. Therefore, GUI programming is done in a textual language, which results in very little separation between the application and interface code. The final code is not reusable and is very difficult to maintain.

To improve reusability and reliability of programming code, more modular architectures such as the object-oriented approach, are being applied. Designing systems using an object-oriented approach involves finding the objects in the problem domain and constructing the relations among the objects. The objects must be structured so that their functionality is assigned in a way that leads to clear delegation of system capabilities that are factored for ease of reuse (Goldberg et al., 1994).

The development of modern interactive applications is also hampered by the lack of integrated modeling techniques and tools. Separate modeling techniques and tools are used during the different stages of development of an application. In addition, many of the problems faced in user interface development are caused by the low abstraction level on which design decisions are made, which again results in lack of overview in the design process (Lauridsen, 1995). Several solutions to these problems have been proposed and include object-oriented programming, visual-programming environments, visual programming languages and visual object-oriented programming.

To make it easier to program GUI's several visual environments for textual programming languages have been developed. Examples include OpenStep (for Objective C), Visual Basic (for Microsoft Basic) and VisualAge (for Smalltalk). These environments integrate tools like graphical editors, visual browsers and visual tracers for debugging. Although it is possible to quickly generate interfaces for small, general applications, any application specific functionality has to be hard coded by the programmer using the textual programming language. It is up to the programmer to map the semantics of the underlying problem domain onto the proper representations in the user interface. It is clear that the developers of these textual programming languages find it very difficult to adapt to the new requirements that are very different from what the languages were originally designed. Existing textual programming languages have been overextended to support the visual and interactive nature of modern graphical interfaces.

Rather than just extend and patch existing textual languages to support the programming of graphical interfaces, several visual programming languages have been developed. These languages have a visual syntax consisting of graphical elements such as pictures, forms and animations (Cox, 1994). A visual syntax may incorporate spatial information such as containment or connectedness, and visual attributes such as location or colour. Most of these languages are graphical such as dataflow languages or state-transition languages. However, the notations of these languages are typically very formal and mathematical in nature and do not easily handle the complexity of real world applications.

Several researchers have tried to combine the advantages of reusability and extensibility of object-oriented design, and the accessibility of visual programming. Their research has resulted in the development of visual object-oriented programming languages. While object-oriented programming has been most successful in the design and implementation of large systems, visual programming has historically been most successful when applied to small programs. However, there is recent academic and commercial interest in component-based programming, whereby existing objects are combined using various visual techniques to build large systems. The functionality and semantics of the objects map to business entities (such as a customer, an invoice, a claims form, etc.), so they represent business objects. These components also extend beyond traditional business objects to include real world artifacts like calendars and clocks. These objects are produced as independent executables that use external messaging mechanisms to communicate. These developments hold the promise of easily building applications and their interfaces from well-defined and well-tested components.

Although the motivation behind the development of visual object-oriented programming seems sound, they pose many problems for expressing and managing large amounts of information, computation and relationships among system elements. These include the following:

1. How to transform an object-oriented model into a GUI;
2. How to integrate modeling tools and visual programming environments;
3. How to ensure that the user interface conforms to HCI guidelines; and
4. How to automate the creation of the user interface.

Several approaches to solving these difficult problems have been proposed. Some provide assistance at the design level (Vanderdonckt, 1993) or by evaluating the produced GUI and making ergonomic proposals or critiques (Lowgren & Nordquist, 1992). Other approaches are concerned with connecting domain analysis to GUI design. For example, UIDE (de Baar et al., 1992) uses a data model, GENIUS (Janssen et al., 1993) uses an entity-relationship model and petri-nets, Mecano (Puerta et al., 1994) uses a domain model; and TRIDENT (Bodart et al., 1995) uses both a modified Entity-Relationship model and a task model as a basis for UI design. However, most of these approaches only look at parts of the design problem.

If the object-oriented approach could be used during the entire development of a new application from the initial analysis to the final interface design, many benefits could result. Modeling techniques and tools that support the object-oriented approach could then be integrated. The underlying semantics of the problem domain could also be easily mapped onto the semantics of interface components. The design of visual interfaces could then use visual tools that support the semantics of the interface elements rather than just their graphical nature, as is current practice.
This project therefore proposes the development of a visual programming environment that would use the rich semantic modeling capabilities of the object-oriented approach during the entire development of an application. The environment will support:

1. The interactive definition of object types and functionality;
2. The interactive management of these reusable objects to build applications;
3. The interactive definition of an object's visual display; and
4. The use of stored ergonomic knowledge to generate a user interface.

The proposal is to develop the environment to look and work very similar to the interface of Visual Basic, but to provide support for the development of applications and their interfaces on a much higher level of abstraction. The environment will be implemented in the Java programming language. Java has quickly become a standard as a cross-platform technology. Java supports component technology in the form of Java Beans. Java Beans provide a standard application programming interface (API) that can be extended to support the development of robust business objects.

The environment will support the design of a visual data dictionary through non-traditional methods. Traditionally data models are designed using formal programming languages or explicitly within a DBMS using a DDL. This environment, however, proposes to support the visual definition of the data dictionary through the interactive definition of class types. A new class is added to the schema by manipulating visual elements to construct a graphical template from which a new class type is generated. The environment would provide a toolbar of defined classes that could be visually manipulated and grouped to define new classes.
The environment will further distinguish between concrete interaction objects (CIO) and abstract interaction objects (AIO). A CIO is a real interface element that the user can manipulate such as a push button or list box. AIO's consist of abstractions of all CIO's from both presentation and behaviour points of view. The abstraction of the user interaction provides several significant advantages. Firstly, a high-level description of the user interface such as that provided by an abstract interface model allows the designer to reason at a level of abstraction

abstraction removed from implementation details, focusing on the behaviour and semantics of the interface rather than the interaction details. Secondly, it takes account of existing user interface design guidelines. Thirdly, it is easier to provide tool support for the process when it is decomposed into a number of sub-activities, each with their own concerns and associated guidelines.

The programmer would therefore design the data dictionary by gathering AIO's on a visual form. Each of these AIO's can be interactively customized to represent the semantics of the underlying concepts in the problem domain. The mapping of the AIO's onto CIO's will be performed by the system based on the declared semantics of the schema information and criteria like available screen space. The system would present the programmer with a list of possible CIO's from a repository of interface objects. The resulting set of forms could serve as the conceptual model of the underlying application.
To help the designer with the eventual interface design, the environment will support several expert systems that use the semantics of the AIO's. These include tools to validate the interface design against HCI guidelines for issues like colour and visual layout.

1.4. Our role in this project

The part of the VEROD project we worked on was the mapping of the AIO's onto CIO's. We have been asked to provide the VEROD project with a set of rules for the selection of the interaction objects (IOs) as well as with a complete library of CIOs implemented in Java.

2. Problem statement

2.1. Definition of the problem: The selection of IOs

The problem we had to solve was the question of which interaction objects should be used in each particular case. Ergonomic rules for selecting IOs can be found in style guides, standards, design guides or some empirical studies. Here follows a description of the problem.

2.2. Knowledge-Based Techniques *(Vanderdonckt, 1995)*

A first prototype of the supporting system was grounded on several *algorithms* for selecting IOs from a series of functional and operational specifications (e.g., data type, data length, number of possible values, domain definition, expandability by user). As these algorithms were completely imbedded within the system, they were rendered completely opaque. As ergonomic rules were coded by algorithms that vary from one data type to another, they were virtually unmaintainable.

A second version introduced a *set of selection rules* contained in an expert system. There was a clear need for identifying each possible value for each specification to avoid multiple conditions (e.g., date type = alphanumeric AND date type <> integer OR....., THEN interaction object = list box) that are hard to manipulate by designers. The system was able to select a single IO out of twenty-five different IOs from seventeen parameters for nine supported data types. Therefore, the production rules have been made canonical, leading to a better understanding and customization of rules, but also leading to three shortcomings:

1) Rule redundancy : the canonization of production rules implies redundancy because a same ergonomic rule can be found at different situations;
2) Lack of visibility and follow-up: one salient feature of expert system is the ability to fully explain their reasoning. Executing production rules textually is not very representative for designers;
3) Exceeding of specification work: the more specifications are provided, the more appropriate the selected IO will be. But designers dislike to be forced to specify all details of each information or action before the automatic selection. It is too constraining and unrealistic.

2.3. Discussion *(Vanderdonckt, 1995)*

Whereas rule redundancy seems impossible to avoid, the two last problems have been addressed as follows:

1) The lack of visibility of production rules invited us to order rules into *decision tables* that can be graphically represented by *decision trees* techniques. The production rules in the expert system do not need to adhere to particular execution order since the flow of control is not determined by the order in which the selection rules have been coded. This is no longer the case with decision tables and decision trees. Decision trees illustrate search in a

state-space representation where a preferred order of rule processing is imposed. This order might not be optimal in all cases. But the fact that the selection for an appropriate IO through a decision tree can be illustrated graphically is far most appreciated by designers. Each node represents a state where a current IO is selected, and the links represent a change from this state to another representing a more ergonomic IO. This change results from this examination of the current value of the information specification.

2) This version of selection was rather automatic and straightforward since an IO was selected from all the specification contained in a repository. Designers' attitude was passive. To make them more participative, it was suggested not to force them to input all required specifications before generation. Instead, they provide minimal specification for each information. From this, the software automatically selects a first proposal. According to the current state in the decision tree, the software becomes more interactive by asking designers one question at a time. The different possible answers, corresponding to the allowed future paths in the decision tree, are presented.

When the designer provides an answer, not only a more ergonomic IO is selected but the corresponding specification is added in the repository. Designers are more likely to see the direct impact of multiple specifications. Therefore, the decision tree could be explored with forward chaining, backward chaining, or bi-directional chaining. As long as the designer wants to proceed, a more ergonomic IO is selected.

Graphical representation of decision trees combined with flexibility implied by directional and progressive chaining offer a truly computer-aided selection of IOs which seems really active by establishing explicit control by designers.

Another interesting feature included in the computer-aided selection of interaction objects is its independence across multiple computing platforms. Indeed, rather than selecting particular CIOs, the system is based on an abstraction of behaviour (AIO), leaving presentation aspect outside. Thus, decision rules contained in the decision tree no longer work on particular instances of interaction objects belonging to different environments, but rather select an AIO.

3. Requirement Analysis

3.1. Introduction

In order to solve the problem (see 2.1), our project could be split into two different linked problems:

1) The first problem was to provide a set of rules to select IOs.
2) The second problem was the development of a library of CIOs containing all the components involved as an output into the selection rules.

3.2. Selection of user interface components

The problem we had to solve was the question of which interaction objects should be used in each particular case. In order to make the right decision we need a matching mechanism between needs and components.

For instance, if the user needs a component to describe the "Name" attribute of a person, this particular attribute can be described with a few criteria like

- the type of this attribute,
- the domain of values of this attribute,
- the number of values in the attribute.

In this case, the type of the value of the attribute is an *alphanumeric* value. The domain is *unknown* and the number of values in the attribute is *1*. Giving those 3 criteria, the selection mechanism will be able to identify that the most suitable IO for this particular attribute will be an Edit box (see Figure 3.5) allowing the user to type in an alphanumeric value into a text field.

3.3. Library of components

3.3.1. Low Level Components

Most Integrated Development Environment (IDE) (Delphi, Visual Basic,...) provides the user with user interface components (UI components) that are seldom used in isolation. Sometimes, the user will have to combine many of those components in order to make a single useful IO.

For instance, if a designer needs a UI component that allows the user to type in his surname, he will need to combine the following components:

1) A *label* (Figure 3.1) in order to explain to the user what is the value expected.

Figure 3.1 : A label

2) A *text field* (Figure 3.2) in which the user will be able to type in his surname

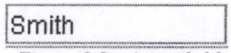


Figure 3.2 : A text field

3) A *panel* (Figure 3.3) in which the designer will add the label and the text field if he wants those two components to move simultaneously

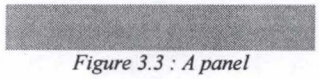


Figure 3.3 : A panel

Moreover, the designer will have to explicitly define the mnemonic relationship between the label and the text field to allow the user to give the focus to the text field by typing the "ALT+ mnemonic" key combination.

We will call all those components that need to be combined in order to produce a single usable IO, the *Low-Level Components.* They are *non-final components*, which means that they can't be used in isolation.

3.3.2. High Level Components

3.3.2.1. Definition

What we decided to call the *High Level Components* are *final components* opposed to the *Low Level Components* which are *non-final components*. That means that the user can include them directly into a form. They have simple methods and properties that make them easy to use. When working with High Level Components, the user will have to write minimal code to finally get complete and useful components. The High Level components are the combination of Low Level Components, which means that the entire interaction between those components is already implemented inside the High Level Components.

For instance, if a designer needs a component that allows the user to type in his surname, he won't need to combine himself Low Level Components and will simply add to his form the Edit Box High Level Component. (see Figure 3.4 below)

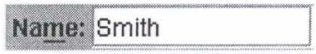


Figure 3.4. : An Edit Box

3.3.2.2. Description of the High Level Components

3.3.2.2.1. Introduction

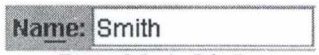
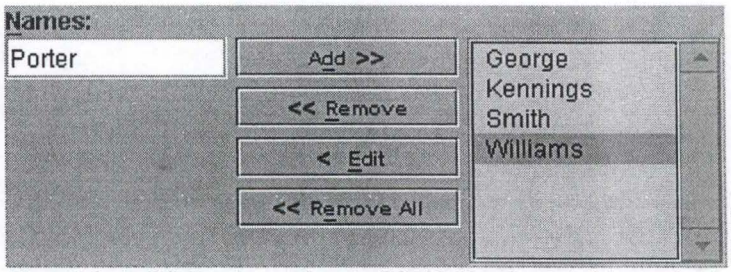
The following tables contain a short description of most of the High Level Components that will be used later on this report. Those components described are AIOs, which means that the pictures are only indicative of what could be the graphical representation of the component.

We decided to sort these components according to the type of the values that they handle.

The different categories are:

1. Non-specific components (see Table 3.1)
2. Integer components (see Table 3.2)
3. Date components (see Table 3.3)
4. Time components (see Table 3.4)
5. Boolean components (see Table 3.5)
6. Graphical components (see Table 3.6)

3.3.2.2.2. Non-specific components

Definition	Graphical representation
An E*dit Box* is a rectangular control in which the user enters or edits a single line text.	 *Figure 3.5: An Edit Box*
An *Edit Box Accumulator (or Editable Non-Contextual Accumulator)* is the combination of an Edit Box (see Figure 3.5 above) with an Accumulator. It allows the user to select an undefined number of text values if the domain of values is unknown.	 Figure 3.6 : An Edit Box Accumulator (or Editable Non-Contextual Accumulator)

A *Multi-Line Edit Box* is a rectangular control in which the user enters or edits a multiple-line text.

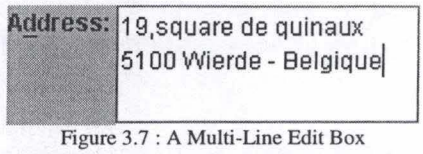


Figure 3.7 : A Multi-Line Edit Box

A *Multi-Line Edit box Accumulator* is the combination of a Multi-Line Edit Box (see Figure 3.7 above) with an Accumulator. It allows the user to select an undefined number of multi-line text values if the domain of values is unknown.

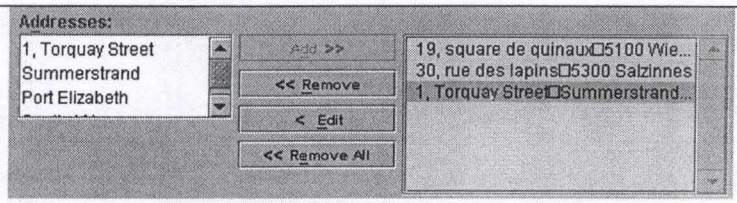


Figure 3.8: A Multi-Line Edit Box Accumulator

A *List Box* is a convenient, preconstructed control for displaying a list of choices for the user. The choices can be either text, icons, or both. The purpose of a list box is to display a collection of items and, in most cases, support selection of a choice of an item or items in the list.

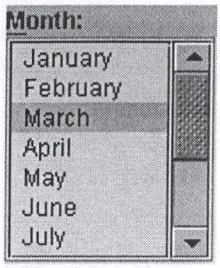


Figure 3.9: A List Box

A *Scrollable List Box* is a List Box (see Figure 3.9 above) with a "fast" scrollbar allowing a much faster scrolling.

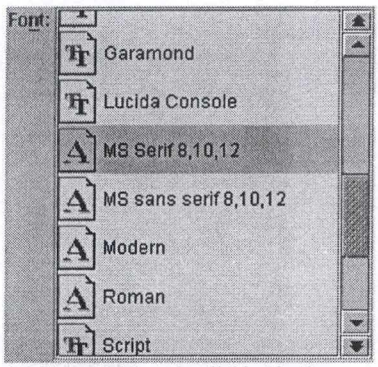


Figure 3.10: A Scrollable List Box

Like a single selection List Box (see Figure 3.9 above), a *Drop-Down List Box* provides for the selection of a single item from a list of items; the difference is that the list is displayed upon demand. In its closed state, the control displays the current value for the control. The user opens the list to change the value

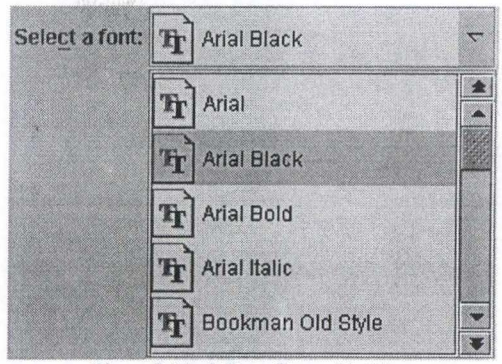


Figure 3.11: A Drop-Down List Box

A *List Box Accumulator (or Non-Editable Contextual Accumulator)* is a control that allows the user to select one or more than one value among the values displayed into a List Box (see Figure 3.9 above). The first list is called the "values list" while the second list is called the "selected values" list. If the user wants to add an item to the selection, he selects an item into the values list and clicks on the "add" button. If the user wants to remove an item from the selection, he selects an item into the selected values list and clicks on the "remove" button. The "add all" and "remove all" buttons allows the user either to select all the values or to deselect all the selected values.

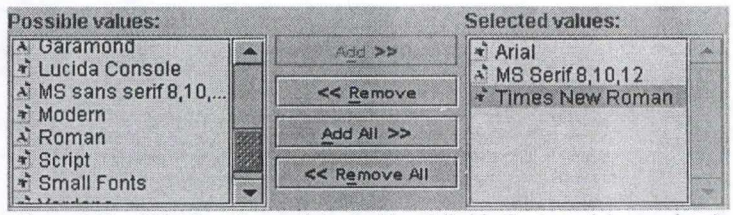


Figure 3.12 : *A* List Box Accumulator (or Non-Editable Contextual Accumulator)

A *Combo Box* is a control that combines a text field with a List Box (see Figure 3.9 above). This allows the user either to type in an entry or to choose one from the list.

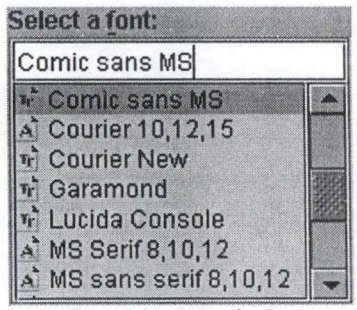


Figure 3.13: A Combo Box

A *Drop-Down Combo Box* combines the characteristics of an Edit Box (see Figure 3.5 above) with a Drop-Down List Box

(see Figure 3.11 above). A Drop-Down Combo Box is more compact than a regular Combo Box (see Figure 3.13 above); it can be used to conserve space, but requires additional user interaction to display the list.

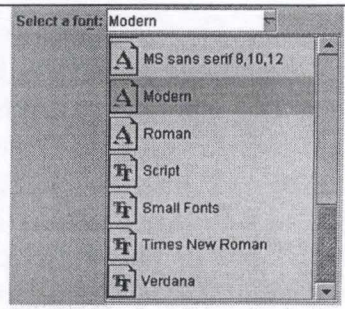


Figure 3.14: A Drop-Down Combo Box

A Combo box Accumulator (or Editable Contextual Accumulator) is the combination of a Combo Box (see Figure 3.13 above) with an Accumulator. It allows the user to select an undefined number of text values if the domain of values is mixed. That means that some of the possible values are known, but that the user can also add new values.

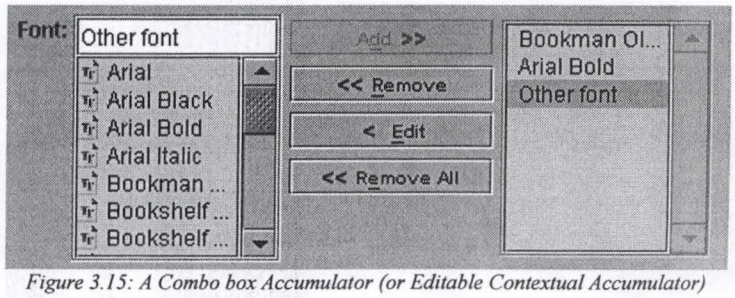


Figure 3.15: A Combo box Accumulator (or Editable Contextual Accumulator)

A *Group Box* is a container that groups components sharing a same informational structure. A Title Label can be used to inform the user on the content of the Group Box.

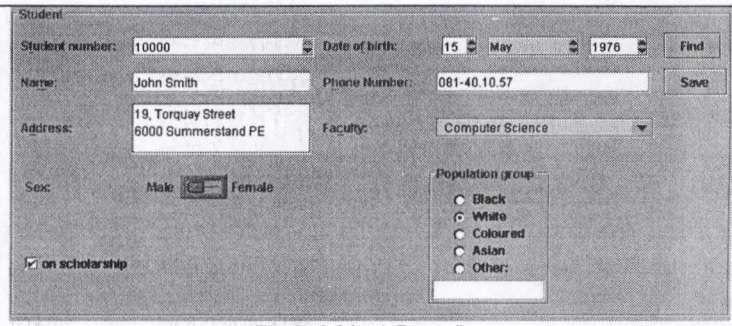


Figure 3.16 : A Group Box

A *Radio Group* is a group of *Radio Buttons*. A Radio Button is an item that can be selected or deselected, and which displays its state to the user. In a Radio Group only one button at a time can be selected.

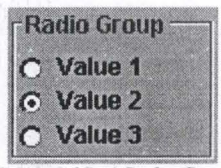


Figure 3.17: A Radio Group

A T*ext Radio Group* is a Radio Group (see Figure 3.17 above) that allows the user to choose either one of the displayed values or to edit another value in an Edit Box. If the 'other' value is unselected, the Edit Box is disabled.

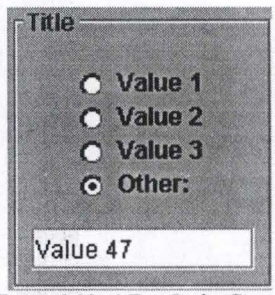


Figure 3.18: A Text Radio Group

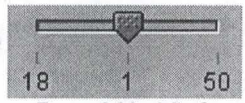
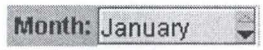
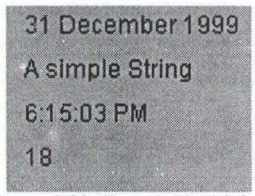
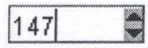
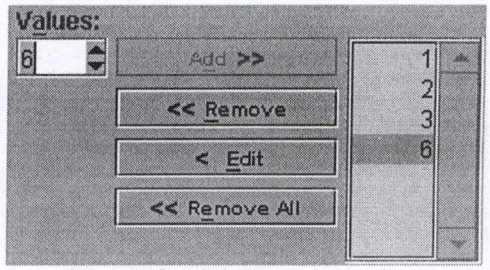
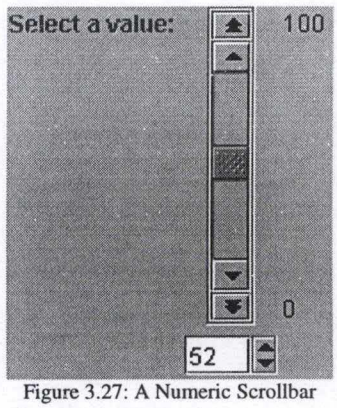
A *Scale* is a component that lets the user graphically select a value by sliding a knob within a bounded interval.	 *Figure 3.20: A Scale*
A *Table* is a component that presents data's in a two-dimensional table format.	Babylon 5 StarShips: Picture / Name / Race / Crew / Jump Point Capable Shuttle / Earth Alliance / 2 / ☐ Warlock Destroyer / Earth Alliance / 250 / ☑ G'Quan Cruiser / Narn Regime / 100 / ☑ *Figure 3.21: A Table*
A *Unitary List Box* is a text box that accepts a limited set of discrete input values that can make up a circular loop. The user can't type a text value directly into the control but he can use the buttons to increment or decrement the value.	 Figure 3.22: A Unitary-List Box
A *Masked Label* is a display area that allows the user to display values from different types such as Date, Time, Integer and String.	 *Figure 3.23: Four* masked *labels*
A *Multi-Line Label* is a display area that allows the user to display a multi-line String.	 *Figure 3.24: A Multi-*Line *Label*

Table 3.1: Non-specific components

3.3.2.2.3. Integer components

Definition	**Graphical representation**
A *Spin Button* is a text box that accepts a limited set of discrete ordered input values that can make up a circular loop. The user can type a text value directly into the control or use the buttons to increment or decrement the value.	 Figure 3.25: A Spin Button
A *Spin Button Accumulator* is the combination of a Spin Button (see Figure 3.25 above) with an Accumulator. It allows the user to select an undefined number of numeric values if the domain of values is unknown	 Figure 3.26: A Spin Button Accumulator
A *Numeric Scrollbar* is a control that allows the user to select a numeric value either by scrolling the scrollbar or by typing the value into the Spin Button (see Figure 3.25 above).	 Figure 3.27: A Numeric Scrollbar

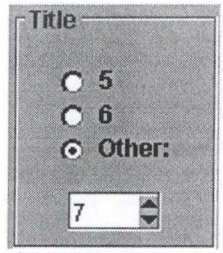
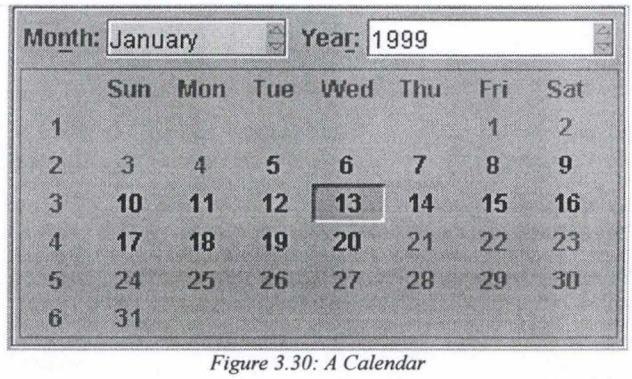
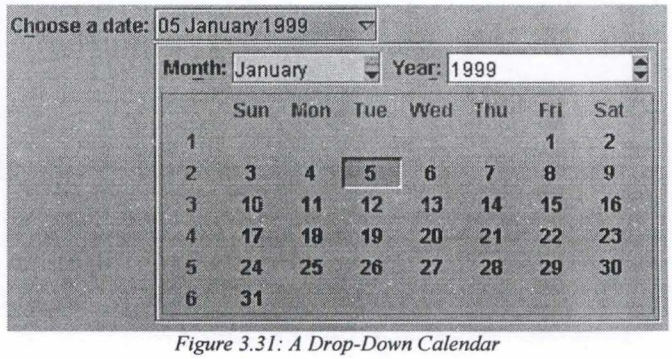
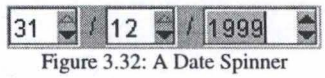
A *Numeric Radio Group* is a Radio Group (see Figure 3.17 above) that allows the user to choose either one of the displayed values or to select another value in a Spin-Button (see Figure 3.25 above). If the 'other' value is unselected, then the Spin-Button is disabled.	 *Figure 3.29: A Numeric* Radio *Group*

Table 3.2: Integer components

3.3.2.2.4. Date components

Definition	**Graphical representation**
A *Calendar* is a control that allows the user to select either one or a group of dates.	 *Figure 3.30: A Calendar*
A *Drop-Down Calendar* is a control that allows the user to select one date. It is better to use a Drop-Down Calendar instead of a Calendar (see Figure 3.30 above) when the screen density is high. It works just like a Drop-Down List Box (see Figure 3.11 above) except that it is a calendar that is displayed when the user clicks on the button.	 *Figure 3.31: A Drop-Down Calendar*
A *Date Spinner* is a component that allows the user to select a date value.	 Figure 3.32: A Date Spinner

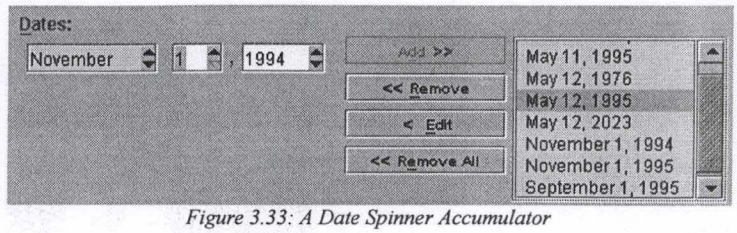
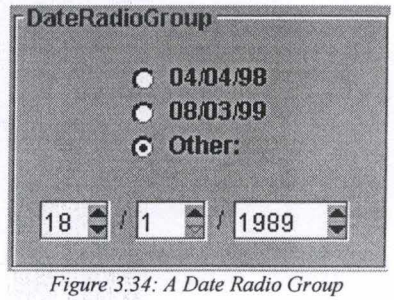
A *Date Spinner Accumulator* is the combination of a Date Spinner (see Figure 3.32 above) with an Accumulator. It allows the user to select an undefined number of dates if the domain of values is unknown.	 *Figure 3.33: A Date Spinner Accumulator*
A *Date Radio Group* is a Radio Group (see Figure 3.17 above) that allows the user to choose either one of the displayed values or to select another value in a Date Spinner (see Figure 3.32 above). If the 'other' value is unselected, then the Date Spinner is disabled.	 *Figure 3.34: A Date Radio Group*

Table 3.4: Date components

3.3.2.2.5. Time components

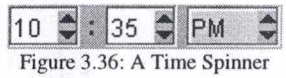
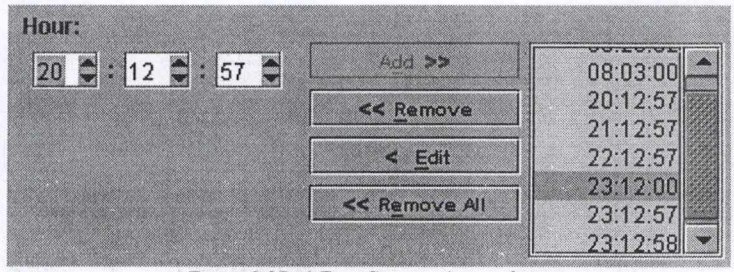
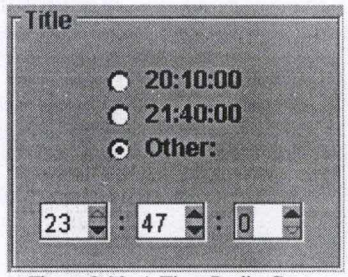
Definition	**Graphical representation**
A *Time Spinner* is a component that allows the user to select a time value.	 Figure 3.36: A Time Spinner
A *Time Spinner Accumulator* is the combination of a Time Spinner (see Figure 3.36 above) with an Accumulator. It allows the user to select an undefined number of time values if the domain of values is unknown.	 *Figure 3.37: A Time Spinner Accumulator*
A *Time Radio Group* is a Radio Group (see Figure 3.17 above) that allows the user to choose either one of the displayed values or to select another value in a Time Spinner (see Figure 3.36 above). If the 'other' value is unselected, then the Time Spinner is disabled.	 Figure 3.38: A Time Radio Group

Table 3.5: Time components

3.3.2.2.6. Boolean components

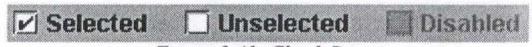
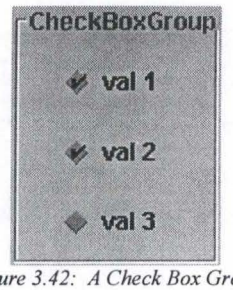
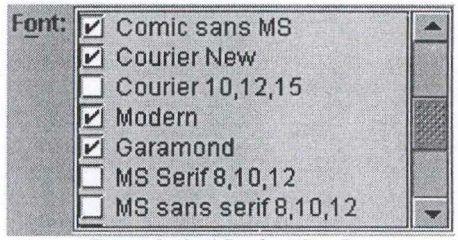
Definition	Graphical representation
A *Switch* is a Component that allows the user to choose between two opposites values	 *Figure 3.40: A Switch*
A *Check Box* is an item that can be selected or deselected, and which displays its state to the user.	 *Figure 3.41: Check Boxes*
A *Check Box Group* is a group of Check Boxes (see Figure 3.41 above). In a Check Box Group more than one button can be selected at a time.	 *Figure 3.42: A Check Box Group*
A *Boolean List Box* is a List Box (see Figure 3.9 above). that contains Check Boxes (see Figure 3.41 above). It is useful if the number of Check Boxes needed is too big.	 *Figure 3.43: A Boolean List Box*

Table 3.6: Boolean components

3.3.2.2.7. Graphical components

Definition	Graphical representation
A *Static Icon* allows the user to display a picture.	 *Figure 3.44: A Static Icon*

A *Browse Button* is a control that allows the user to select a graphical value. It is the combination of a push button and a Static Icon (see Figure 3.44 above). When the user clicks on the button, an open file dialog box pops up and the user can select a file.

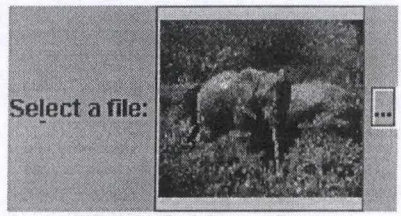


Figure 3.45: A Browse Button

A *Browse Button Accumulator (or Editable Non-Contextual Graphical Accumulator)* is the combination of a Browse Button (see Figure 3.45 above) with an Accumulator. It allows the user to select an undefined number of graphical values if the domain of values is unknown.

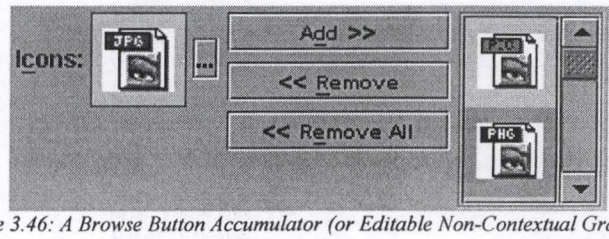


Figure 3.46: A Browse Button Accumulator (or Editable Non-Contextual Graphical Accumulator)

A *Graphical Combo Box* is a Combo Box (see Figure 3.13 above) that is used to select a graphical value. To select a value, the user can either use the Browse Button (see Figure 3.45 above) to select a file or select one of the values of the list.

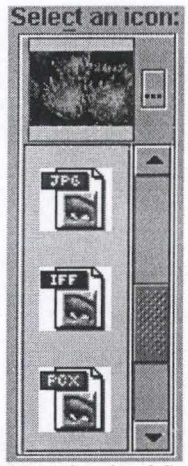


Figure 3.47: A Graphical Combo Box

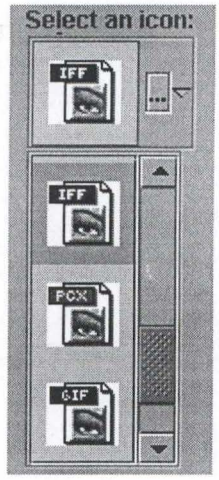
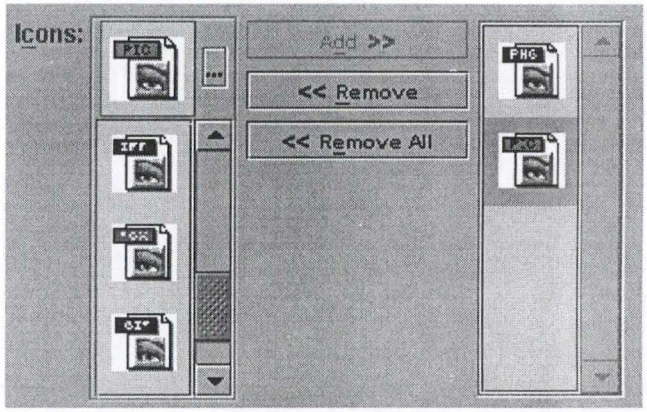
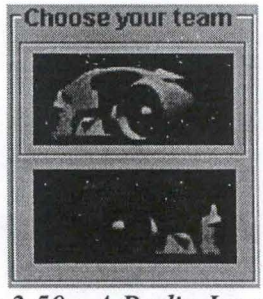
A *Drop-Down Graphical Combo Box* is a control that allows the user to select a graphical value. It is better to use a Drop-Down Graphical Combo Box instead of a Graphical Combo Box (see Figure 3.47 above) when the screen density is high. It works just like a Drop-Down List Box (see Figure 3.11 above) except that the user has the possibility to select a value that is not in the list by using the browse button.	 Figure 3.48: A Drop-Down Graphical Combo Box
A *Graphical Combo box Accumulator (or Editable Contextual Graphical Accumulator)* is the combination of a Graphical Combo Box (see Figure 3.47 above) with an Accumulator. It allows the user to select an undefined number of graphical values if the domain of values is mixed. That means that some of the possible values are known, but that the user can also add new values.	 *Figure 3.49:* A *Graphical Combo box Accumulator (or Editable Contextual Graphical Accumulator)*
A *Radio Icon Group* is a group of Graphical *radio buttons*. A Radio Button is an item that can be selected or deselected, and which displays its state to the user. In a Radio Icon Group, only one button at a time can be selected.	 *Figure 3.50 : A Radio Icon Group*

Table 3.7: Graphical components

3.3.2.3. Advantages of the High Level Components

Providing the user with a library of high level components instead of a library of Low Level Component presents the following advantages:

1) As combined components they have some simplified layout managers. For instance, the label of an Edit Box can be placed either above or on the left of the text field according to the wish of the user. (see Figure 3.52 and 3.53 below)

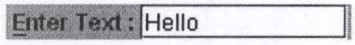


Figure 3.52 : An horizontal Edit Box

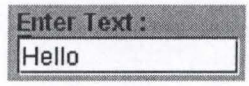


Figure 3.53 : A vertical Edit Box

2) Some of these High Level Components are not available in most of the existing IDE. Therefore the designers often use a less suitable UI component instead of the most appropriate one. Most of the time, those components are unknown to the user. One of the best illustrations is the List Box Accumulator (see Figure 3.12).

3) The interaction between all the combined Low Level Components is already implemented inside the High Level Component. For instance, the List Box Accumulator component (see Figure 3.12) implements the entire interaction between the buttons and the List Boxes. For instance, if the user clicks on the Add button, the selected value in the first List Box will be added to the second one.

All these characteristics lead to the conclusion that the High Levels Components can give to their user a valuable gain of time and give him the opportunity to use more suitable components. This is why we decided that the library of components that we will develop would consist of High Level Components.

4. Analysis and design

The problem can be split into two main stages. The first one is the definition of the selection rules used in the implementation of the selection algorithm. The second one is the design of all the components.

4.1. Design of the selection algorithm

4.1.1. Selection trees

For all the reasons previously explained (see 2.3), the technique we decided to use to implement a selection algorithm is the technique of the *selection trees*.

The Figure 4.1 below represents a selection tree used for the selection of AIOs.

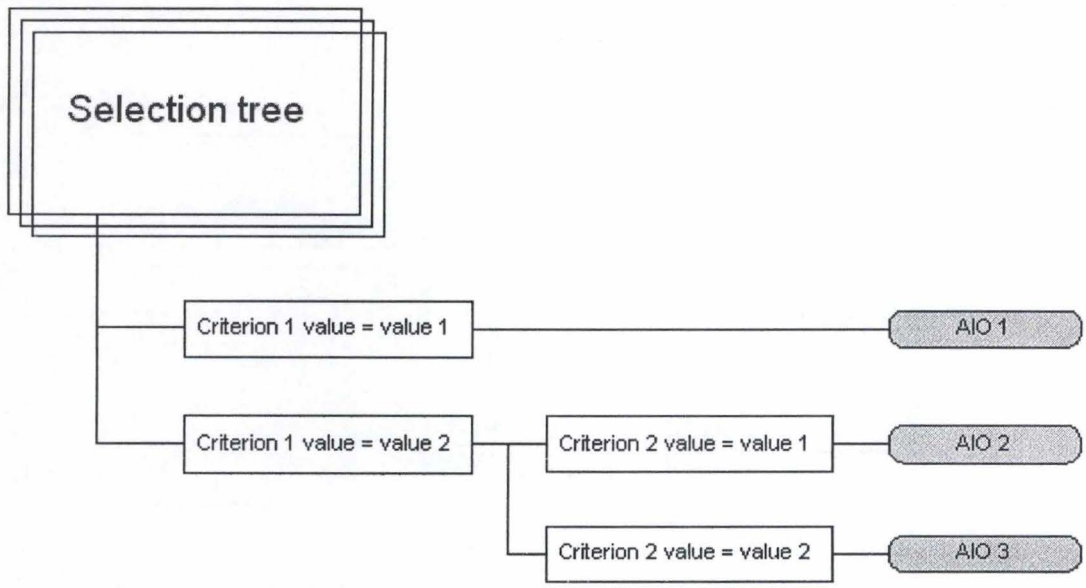


Figure 4.1: A selection tree

A selection tree consists of *nodes* and *leaves*. A definition of those concepts follows.

1) **The nodes**

A *node* is a decision point where the tree is divided into several sub-trees. (see Figure 4.2 below). When reaching a node, the user has to decide what is the value of the criterion associated with the node, and then go through the sub-tree connected to this particular value.

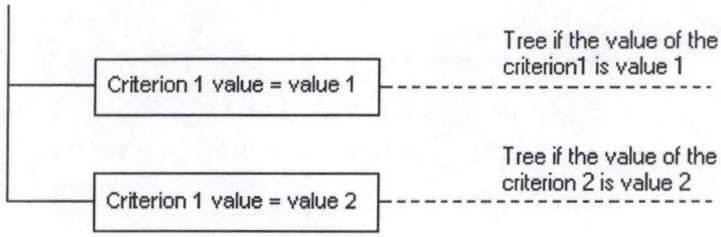


Figure 4.2 : A node

For instance, if the value of the criterion 1 is equal to value 2 (see Figure 4.3 below), then the upper part of the tree is irrelevant and the user can go trough the lower part of the tree.

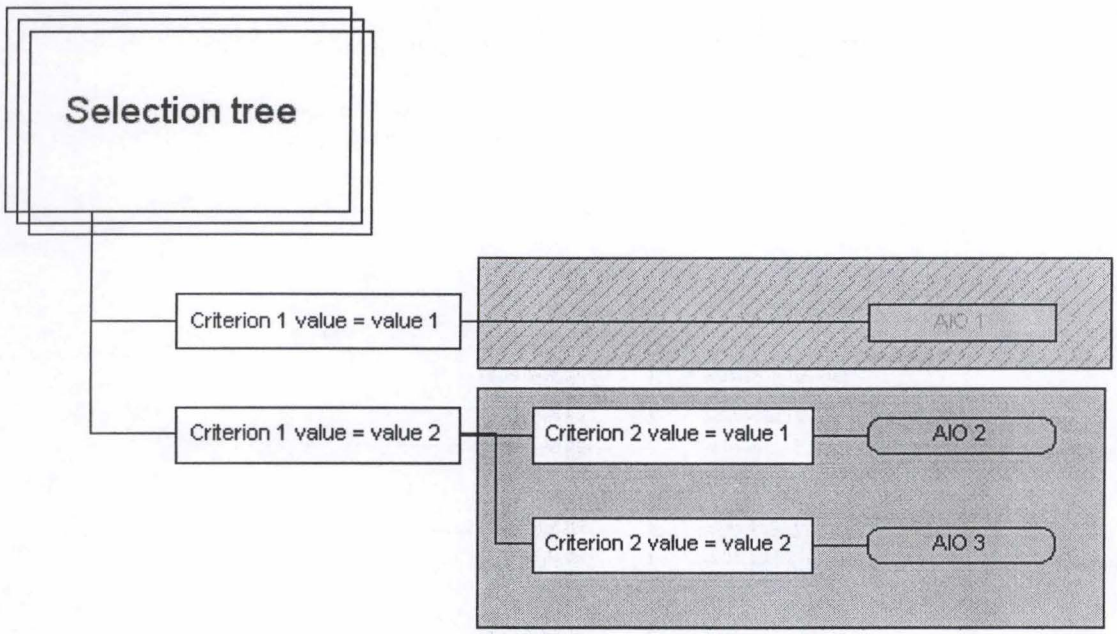


Figure 4.3 : A selection tree

2) **The leaves.**

A *leaf* represents an AIO (see Figure 4.4 below). When the user reaches a leaf, it means that the selected AIO for his particular needs is the AIO represented by the leaf.

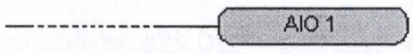


Figure 4.4 : A leaf

4.1.2. Vanderdonckt's selection trees

In order to create rules for the selection of AIOs, we based our research on the selection trees defined by Jean Vanderdonckt in (Vanderdonckt, 1997). Vanderdonckt's selection trees are

defined at the AIO level in order to make this selection problem independent of the target graphical environment. (see 1.2 : AIOs and CIOs)

4.1.2.1. Selection criteria

The selection criteria (that will be the nodes of the trees) he decided to take into account are the following:

1. Interaction type

This criterion defines whether the AIO's goal is to get a value from the user (Input) or to show a value to the user (Display).

The possible values are:

- *Input*
- *Display*

2. Type

This criterion defines what is the type of the values handled.

The possible values are:

- *Date* (e.g. 12 November 1998, 13 April 1256)
- *Time* (e.g. 12:56 , 9:33:25)
- *Graphical* (e.g. a logo, a picture)
- *Integer* (e.g. 5,6,3)
- *Real* (e.g. 5.36, 5.665589)
- *Alphanumeric* (e.g. John Smith, Erdd-65-ssa)
- *Boolean* (e.g. True, False)

3. Domain

This criterion defines the type of the domain.

The possible values are:

- *Known*: The domain is known if all the possible values are known. For instance, the domain for a "Sex" attribute is known because the two possible values (Male and Female) are known.
- *Unknown*: The domain is unknown if all the possible values are unknown. For instance, the domain for a "date of birth" attribute is unknown. If the possible values were defined between two specific dates (e.g. 13 November 1996 and 13 November 1997), then the domain would be known.
- *Mixed*: The domain is mixed if some of the possible values are known but some others are unknown. For instance, the domain for a "city of birth" attribute is mixed because the most current city names are known (e.g. Port Elizabeth, Cape Town, Johannesburg) but it is impossible for the system to know all the existing cities.

4. Choice

This criterion defines whether the user has to select one or more than one value.

The possible values are:

- *Simple* (or Number of values to choose = 1): In this case, the user has to select only one value. For instance, the "date of birth" attribute consists of only one date.
- *Multiple* (or Number of values to choose > 1): In this case, the user can select more than one value. For instance, the "first names" attribute consists of several names (e.g. John Fitzgerald)

5. Number of possible values

This criterion defines the number of values included into the domain.

6. Continuity of the domain

This criterion defines whether or not the domain is continuous.

The possible values are:

- *True*: In this case, the domain is continuous. For instance, if the user has to select a date between January 11, 1999 and January 31, 1999 then the domain is continuous.
- *False*: In this case, the domain is discontinuous. For instance, if the user has to select a date among a list of dates that are unrelated (e.g. {January 11, 1999; February 25, 1995; March 3, 1976;...}), then the domain is discontinuous.

7. Length

This criterion defines what is the maximum length of the possible values.

For instance, the length of the "surname" attribute is 30 because most of the possible values for this attribute are less than 30 characters long.

8. Precision

This criterion defines what is the precision level expected to select a value.

The possible values are:

- *High*
- *Low*

9. Preference for selection

This criterion defines whether the user prefers to select a value with the mouse or if he prefers to type in a value.

The possible values are:

- *True*: In this case, the user prefers to select a value.
- *False*: In this case, the user prefers to type in a value.

10. Opposites values

This criterion defines whether or not the possible values are opposites. It is used only if the type of the values handled is Boolean.

The possible values are:

- *True*: In this case, the possible values are opposites. For instance, if the user has to select whether the temperature of the water is hot or cold, the two possible values (hot and cold) are opposites values.
- *False*: In this case, the possible values are not opposites. For instance, if the user has to select whether the temperature of the water is hot or not, the two possible values are "hot" or it's negation "not hot".

11. Screen Density

This criterion defines whether the screen density is high or low.

The possible values are:

- *High*: In this case, there is not a lot of screen space left.
- *Low:* In this case, there is a lot of screen space left.

12. Orientation

This criterion defines what is the orientation of the AIO.

The possible values are:

- *Vertical*: In this case, the orientation of the AIO will be a vertical one.
- *Horizontal*: In this case, the orientation of the AIO will be a horizontal one.
- *Circular:* In this case, the orientation of the AIO will be a circular one.
- *Undefined*: In this case, the orientation of the AIO will be undefined.

13. Data Type

This criterion defines whether each data is a single value or consists of several different values.

The possible values are:

- *Elementary*: In this case, each data is a single value. For instance, a "name" attribute consists of only one alphanumeric value. (e.g. Smith, Jones, Williams)
- *List*: In this case, each data consists of more than one value. For instance, an "address" attribute consists of a house number, a street, a zip code and a city name. (e.g. 16, Torquay Street, 6000, Port Elizabeth)

14. Data Number

This criterion defines the number of different types displayed into a single piece of information.

The possible values are:

- *1*: In this case, a data consists of several values of the same type. For instance, the "temperatures forecast" attribute for the next week consists of 7 integer values. (e.g. {26,26,25,23,31,32,25})
- *> 1*: In this case, a data consists of several values of different types. For instance, an "address" attribute consists of integer fields for the house number and the zip code, as well as of an alphanumeric field for the street name and the city name. (e.g. 16, Torquay Street, 6000, Port Elizabeth)

15. Number of values to display

This criterion defines whether the AIO must be able to display one or more than one value.

The possible values are:

- *1*: In this case, the AIO is only able to display one value. For instance, the number of values to display for a "daily temperatures of the last week" attribute is 1 because only one week is taken into account. (e.g. {26,26,25,23,31,32,25})
- *> 1*: In this case, the AIO is able to display more than one value. For instance, the number of values to display for a "daily temperatures for all the weeks of last year" attribute is 52 because 52 weeks are taken into account. (e.g. {26,26,25,23,31,32,25}, {27,26,25,23,31,32,25}, {13,26,25,23,31,32,25},{26,26,31,23,31,32,25}, ...)

4.1.2.2. Quadrants

When looking at the whole selection tree, it is interesting to notice that some parts of it are redundant. For instance, the Figure 4.5 and 4.6 below show two identical tree parts.

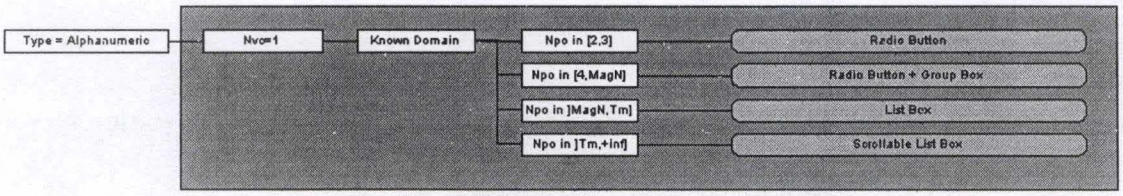


Figure 4.5: a part of the selection tree

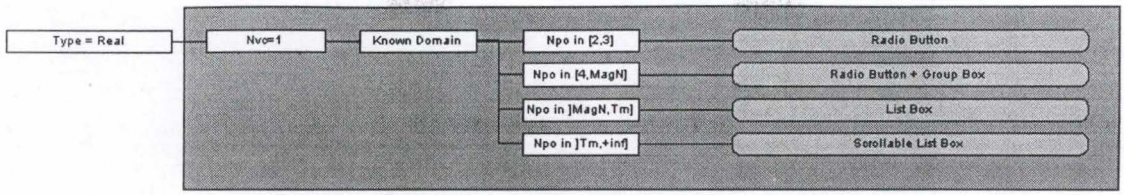


Figure 4.6: another part of the selection tree

In order to simplify the representation of the trees, it is easy to put together all those common parts into distinct selection trees that we will call *quadrants*. (see Figure 4.7 below)

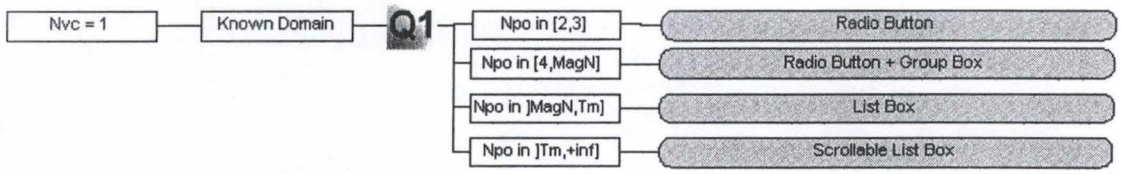


Figure 4.7 : The Q1 quadrant

Six different quadrants have been defined and named Q1, Q2, Q3, Q4, Q5 and Q6. (see Figure A.1 and A.2, Appendix A)

Those common parts will be replaced into the main selection tree by a reference to a specific quadrant. (see Figure 4.8 and 4.9 below)

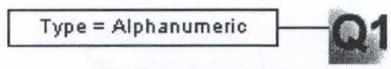


Figure 4.8 : The sub-tree has been replaced by a reference to the quadrant

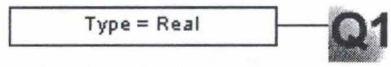


Figure 4.9 : The sub-tree has been replaced by a reference to the quadrant

4.1.2.3. Splitting of the selection tree

The size of the entire tree being quite considerable, we decided to split it into 8 different selection trees (named D1, D2, ..., D8) according to the type of the values handled. The Figure 4.10 below shows the *top-level* selection tree.

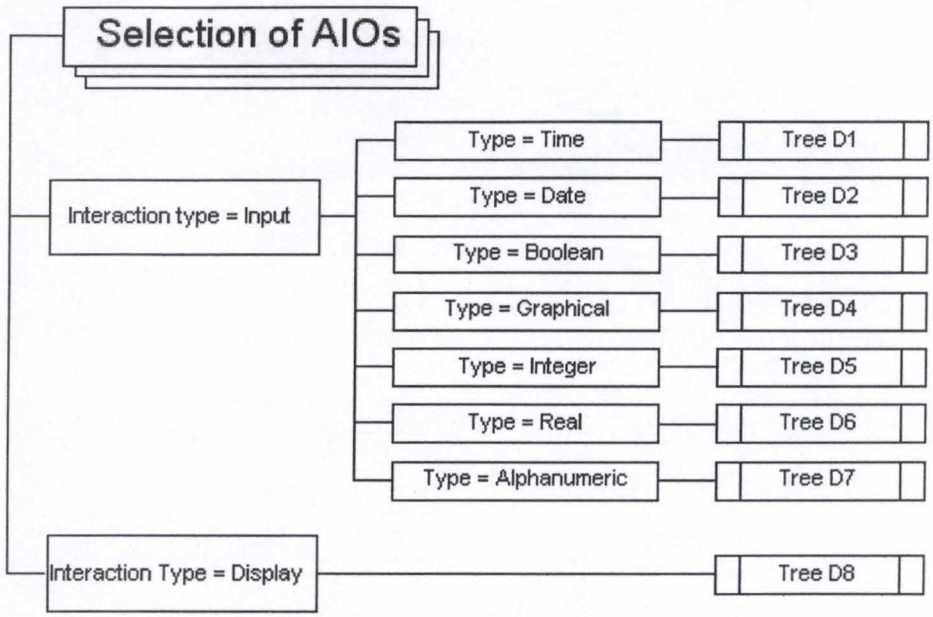


Figure 4.10 : the top-level selection tree

The individual selection trees are displayed in the Figure A.3 to A.11 (Appendix A).

4.1.2.4. Abbreviations used in the trees

The abbreviations used in the selection trees are contained in the Table 4.1 below.

MagN	Magical number (= 7 +- 2 according to the experience level of the user)
Bn	Maximum number of distinguishable values on a Scroll Bar, Scroll Cursor or a Scale (= +- 10)
Lm	Maximum length of an alphanumeric item (= +- 40)
Tm	Maximum number of items in a list (= +- 50)
Npo	Number of possible values
Nvc	Number of values to choose
Nvd	Number of values to display

Table 4.1: Abbreviations used in the selection trees

4.1.3. Adaptation of Vanderdonckt's rules

We decided to adapt Vanderdonckt's trees for two reasons. First because his trees contain too many AIOs. If we wanted to develop efficiently a library of UI components, we had to limit ourselves in the number of components implemented. Secondly, because a few criteria didn't seem appropriate to us and because some AIO's choices could be improved. These will be discussed below.

The individual simplified selection trees are in the Appendix B .

4.1.3.1. Modifications

The modifications that we made are the following:

1) Vanderdonckt's trees contain the *screen density* criterion that doesn't seem suitable at this stage of the design process. When selecting an AIO, the user is not aware of the number and the size of the other components surrounding this AIO. Therefore the screen density choice will be postponed to a later stage.

In the trees, we always selected the "Low screen density" AIO by default. Ultimately we will add to some of the AIOs an alternative component that takes less screen space.

For instance, the tree in the Figure 4.11 below has replaced the tree in the Figure 4.12.

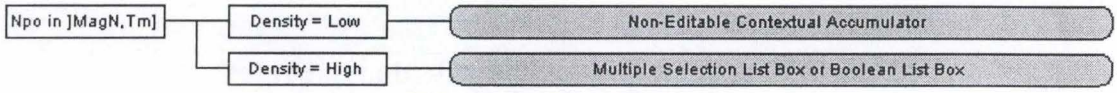


Figure 4.11: Part of Vanderdonckt's selection tree

Figure 4.12: Part of Vanderdonckt's tree without the screen density criterion

2) The *orientation* criterion has been removed from the trees. The reason is that, when selecting an AIO, the user is not aware of the number and the size of the other components surrounding this AIO. Therefore the orientation choice will be postponed to a later stage.

For instance, the tree in the Figure 4.13 below has replaced the tree in the Figure 4.14.

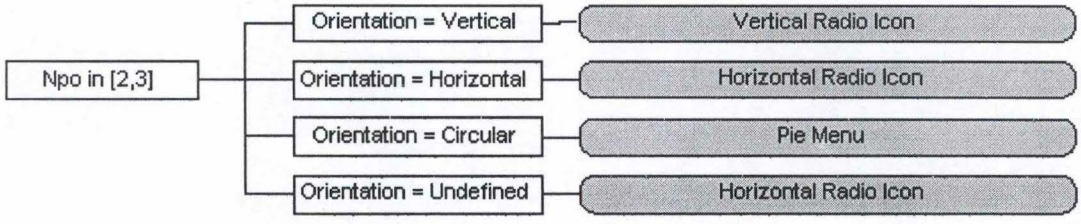


Figure 4.13: Part of Vanderdonckt's selection tree

Figure 4.14: Part of Vanderdonckt's selection tree without the orientation criterion

3) The Time (see Figure A.3) and Date (see Figure A.4selection trees have been considerably changed in order to include components like the Calendar (see Figure 3.30) and the Time Spinner (see Figure 3.36).

4) In some cases, we decided to use more specific AIOs. For instance, instead of using an Edit Box to select one value of the Integer, Date or Time type if the domain is unknown, we use more suitable components like the Spin Button (see Figure 3.25), the Date Spinner (see Figure 3.32) and the Time Spinner (see Figure 3.36). According to the same principle, we used components like the Date Spinner Accumulator (see Figure 3.33), the Time Spinner Accumulator (see Figure 3.37) and the Spin Button Accumulator (see Figure 3.26) instead of the Editable Non-contextual Accumulator (or Edit Box Accumulator) (see Figure 3.6).

The result of this process of simplification was that the number of rules dropped from 256 to130.

4.1.3.2. Selection criteria

The Table 4.2 below compares the criteria used in Vanderdonckt's trees with those used in our simplified trees.

Criterion Name	**Used in Vanderdonckt's trees**	**Used in our trees**
Interaction type	Yes	Yes
Type	Yes	Yes
Domain	Yes	Yes
Choice	Yes	Yes
Number of possible values	Yes	Yes
Continuity of the domain	Yes	Yes
Length	Yes	Yes
Precision	Yes	Yes
Preference for selection	Yes	Yes
Opposites values	Yes	Yes
Screen density	Yes	*No*
Orientation	Yes	*No*
Data type	Yes	Yes
Data number	Yes	Yes
Number of values to display	Yes	Yes

Table 4.2: Comparison of the selection criteria used

4.1.4. Transformation of the selection trees

The next stage was the transformation of the trees into a structure much easier to implement into the VEROD project (see 1.3).We used Excel worksheets to contain and represent the rules in the selection trees.

4.1.4.1. Representation of the criteria

Each criterion will be represented as a *column* into the spreadsheets. Thirteen columns will then represent the thirteen criteria used in our trees (see Table 4.2).

In order to simplify the presentation of the rules, we split the main selection tree into 2 parts at the level of the interaction type criterion. The rules related to the input interaction type will be in the spreadsheet A while the rules related to the display interaction type will be in the spreadsheet B as shown in Figure 4.15 below.

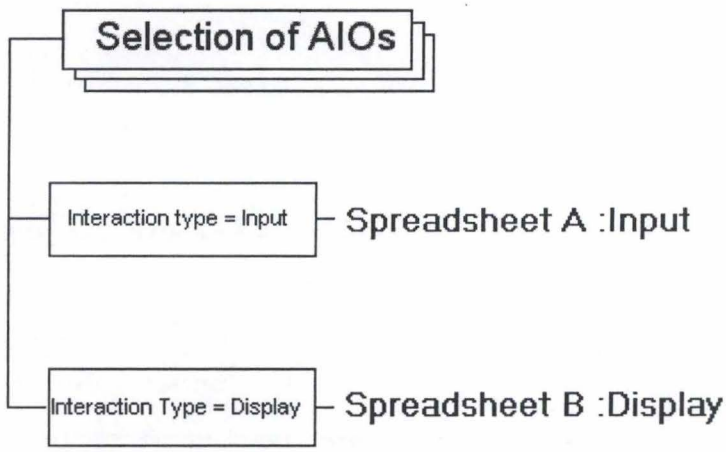


Figure 4.15: The selection tree has been split into two spreadsheets

4.1.4.2. Representation of the rules

Each terminal leaf of the selection trees represents one *rule.* Each node that led to a particular leaf represents a *condition* that led to the selection of a particular AIO. The path followed in the tree in order to reach a particular leaf can identify each particular leaf. (see Figure 4.16 below)

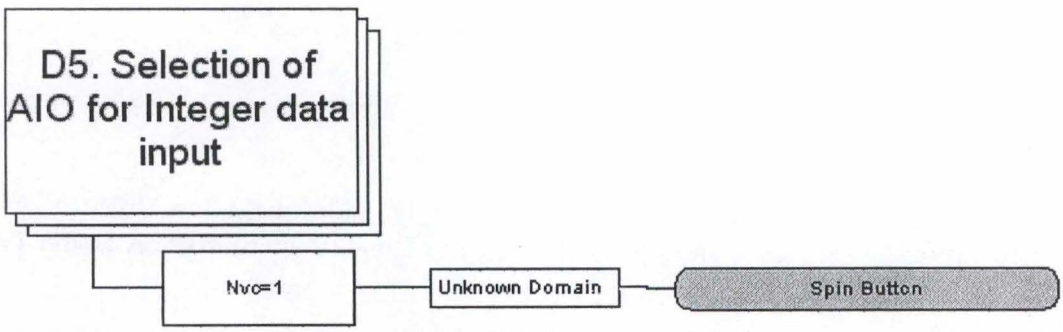


Figure 4.16: The path leading to a particular leaf

For instance, the rule represented by the Figure 4.16 above could be translated into the following formula:

"If the interaction type = input
And if the type = integer
And if the number of values to choose = 1
And if the domain is unknown
Then the selected AIO is a Spin Button"

Or

"(interaction type = input) $\wedge$ (type = integer) $\wedge$ (number of values to choose = 1) $\wedge$ (unknown domain) $\Rightarrow$ selected AIO = Spin Button"

Each rule can then be represented into the spreadsheets as a *row*. Each condition that led to a leaf (or an AIO) will be represented by a specific value in the column of the criterion involved into the condition.

For instance, the rule represented by the tree in the Figure 4.16 above will be presented into the input spreadsheet by the Table 4.3 below.

Selected AIO	Type	Domain	Choice	Npo	Cont	Lg	Precision	Preference for selection	Opposites values
Spin Button	Integer	Unknown	Simple (Nvc=1)						

Table 4.3: A rule into the spreadsheet

The spreadsheets A (input interaction type) and B (display interaction type) are in the Appendix F.

4.1.5. Lower Density Alternatives for some AIOs

The AIOs handled by the selection rules are High Density AIOs. That means that they take up a great deal of space screen. For instance, the user won't have space enough to put 15 List Boxes into a form. Therefore we linked some AIOs with a Lower density AIO.

For instance, the alternative for the List Box (see Figure 3.9) will be the Drop-Down List Box (see Figure 3.11).

Finally the number of AIOs handled by the trees, including the additional alternative AIOs is equal to 92. The list of all the AIOs, along with their Lower Density alternatives is in the Table C.1 (Appendix C).

4.2. Design of the components

4.2.1. Development platform

4.2.1.1. Java

In agreement with our supervisors, we have chosen to use Java as development language for our High Level Components Library. The reasons for this will be discussed below.

Java is a simple, object-oriented, network-enabled, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language. Java makes programming easier because it is object-oriented and has automatic garbage collection. In addition, because compiled Java code is architecture-neutral, Java applications are ideal for a diverse environment like the Internet. (Sun Microsystems 1998,1)

4.2.1.2. Java Beans *(Sun Microsystems 1998,1)*

The Java Beans API makes it possible to write component software in the Java programming language.
Components are self-contained, reusable software units that can be visually composed into composite components, applets, applications, and servlets using visual application builder tools (IDE). That's the main reason why we chose the Bean technology as a development standard for our High-Level Components Library.

Java Bean components are known as Beans. A "Java Beans-enabled" builder tool maintains Beans in a palette or toolbox. You can select a Bean from the toolbox, drop it into a form, modify it's appearance and behaviour, define its interaction with other Beans, and compose it and other Beans into an applet, application, or new Bean. All this can be done without writing a line of code. (see Figure 4.17 below)

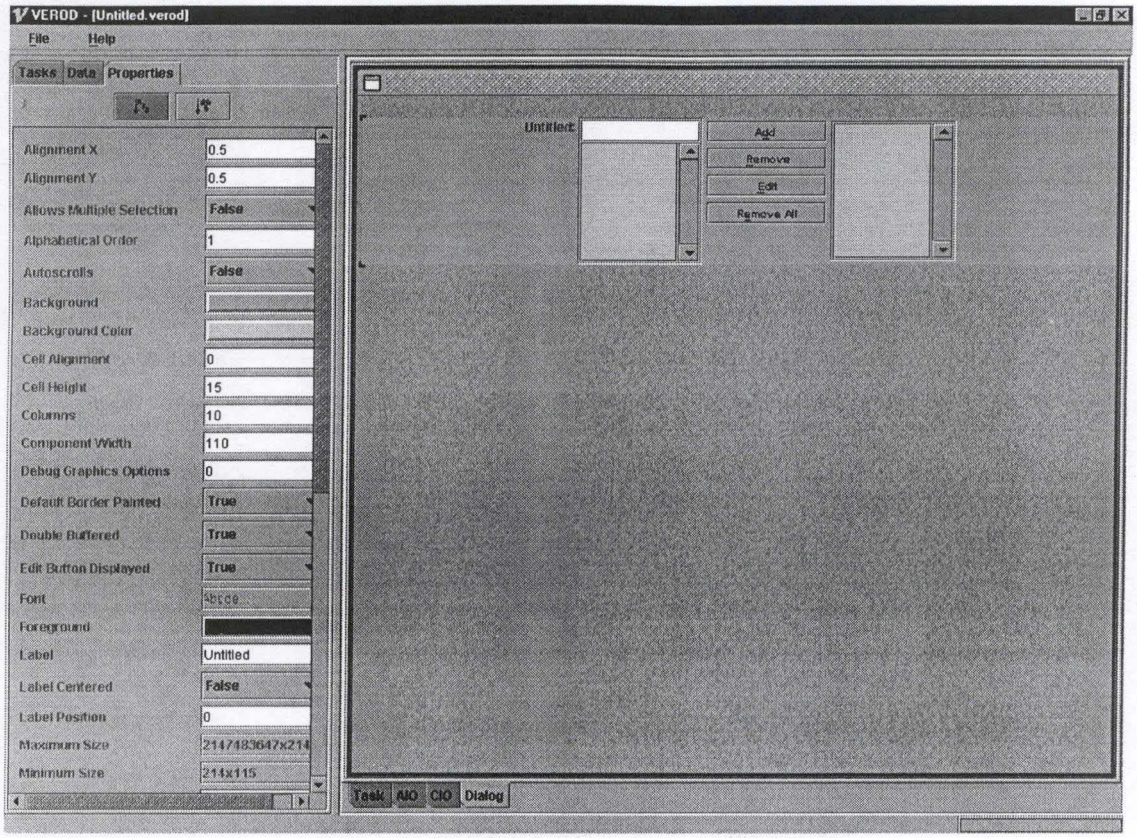


Figure 4.17 : The manipulation of a Beans (from VEROD)

The following list briefly describes key Bean concepts:

- Builder tools discover a Bean's features (that is, its properties, methods, and events) by adhering to specific naming conventions, known as design patterns, when naming Bean features or by explicitly providing property, method, and event information with a related Bean Information class.

- Properties are a Bean's appearance and behaviour characteristics that can be changed at design time. Naming conventions are that for an "X" property, you need a getter method called "getX" and a setter method called "setX" or "isX" is the property is a Boolean value. .

- Beans expose properties so they can be customized at design time. Customization is supported in two ways: By using property editors, or by using more sophisticated Bean customizers.

- Beans use events to communicate with other Beans. A Bean that wants to receive events (a listener Bean) registers its interest with the Bean that fires the event (a source Bean). Builder tools can examine a Bean and determine which events that Bean can fire (send) and which it can handle (receive).

- Persistence enables Beans to save their state, and restore that state later. Once you've changed Bean's property, you can save the state of the Bean and restore that Bean at a later time.

- A Bean's public methods are no different from Java methods, and can be called from other Beans or a scripting environment.

4.2.1.3. Java Foundation Classes

We decided to use The Java Foundation Classes (JFC) to design High-Level Components. The following text on JFC demonstrates why it is the top-choice in GUI design and, therefore, why we chose it. The JFC include many ready-to-use UI components, nicknamed Swing components.

4.2.1.3.1. Swing Architecture (Sun Microsystems 1998,2)

Swing, is a GUI component kit that simplifies and streamlines the development of UI components. Windowing components are the visual components (such as menus, tool bars, dialogs and the like) that are used in graphically based applets and applications.

One the most important feature of Swing components is that they are lightweight. That means that Swing components don't use any platform-specific implementations (which AWT programmers often refer to as "peers."). Instead, Swing creates its components using pluggable look-and-feel (PL&F) modules that are written from scratch and don't use any peer code at all. Consequently, Swing components can typically be incorporated into a program using less code that older "heavyweight" components required. Therefore, Swing components use fewer system resources and produce smaller and more efficient applications than their heavyweight AWT counterparts. Swing has three standard L&F: Metal, Window and Motif.

From an API perspective, the Swing component set extends, but does not replace, the pre-Swing Abstract Windowing Toolkit (AWT). In fact, Swing sits atop part (but not all) of the AWT tool set, as shown in the Figure 4.18 below.

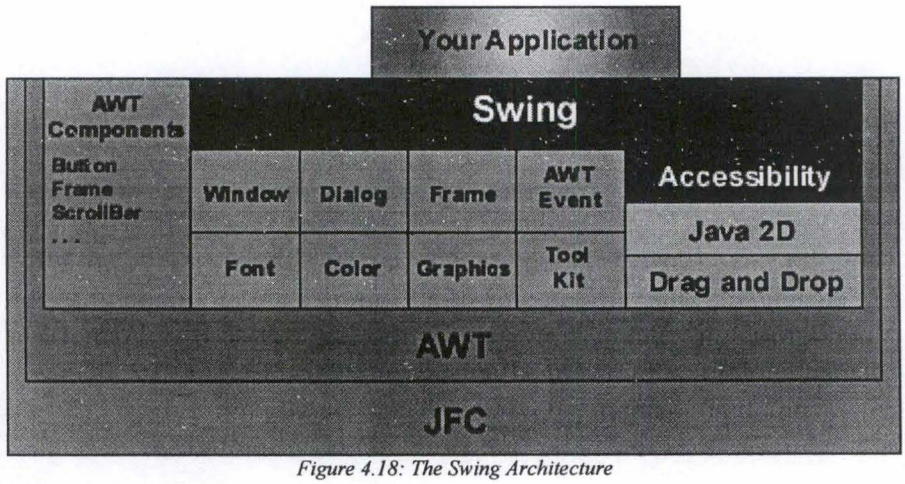


Figure 4.18: The Swing Architecture

The diagram in the Figure 4.18 shows how Swing sits on top of a number of the APIs that implement the various parts of JFC, including the Java 2D and Drag and Drop APIs. Although both those APIs are part of JFC, they are not part of Swing. That's because certain tasks that they perform require use of native code. And Swing components, as we have seen,

never rely on peer code. So Java 2D and Drag-and-Drop appear beneath, not inside, the rectangle labeled "Swing" in the diagram. So does the old-style AWT component set, which is extended (but not replaced) by Swing.

Swing is based on the Model View Controller (MVC) architecture (see Figure 4.19 below). Classic MVC architecture divides each component into three parts: a model, a view, and a controller. In classic MVC architecture, the model manages whatever data or values the component uses, such as the minimum, maximum, and current values used by a scroll bar. The view manages the way the component is displayed. And the controller determines what happens when the user interacts with the component—for example, what occurs when the user clicks a button control.

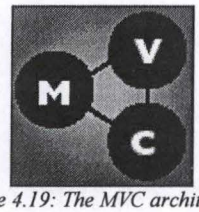


Figure 4.19: The MVC architecture

The Swing architecture is based on a modified MVC design. Classic MVC architecture is simple and elegant, and the Swing team started designing the Swing component set, they experimented with using a pure MVC design. But they soon discovered that classic MVC architecture didn't work very well in real-world Swing application. So they wound up basing Swing architecture on a modified adaptation of the traditional MVC design. To implement Swing components, the Swing team created a modified MVC design. The component design that the Swing team eventually settled on is sometimes referred to a separable *model architecture.*

In Swing's separable model design, the model part of a component is treated as a separate element, just as the MVC design does. But Swing collapses the view and controller parts of each component into a single UI object. The result is an architecture that looks like the one in Figure 4.20.

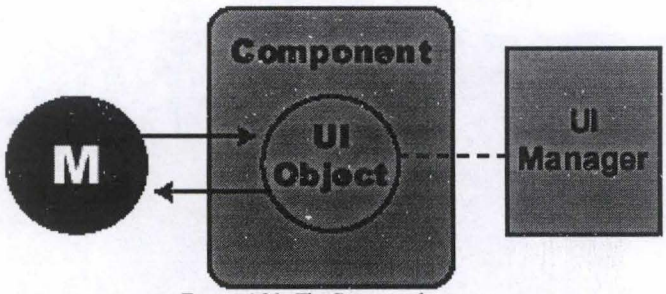


Figure 4.20: The Swing architecture

Swing´s separable model architecture was developed because the creators of Swing found that the traditional MVC design didn't work well in practical terms in Swing-style components. It's mainly because the view and controller parts of a traditional MVC-based component require a tight coupling that is sometimes difficult to achieve in practical terms. For example, traditional MVC architecture makes it very hard to create a generic controller that doesn't know at design time what kind of view will eventually be used to display it.

The UI Manager: To handle the look-and-feel characteristics of its modified-MVC components, Swing defines a class called the UI Manager, which always keeps track of the current look and feel and its defaults. The UI manager, shown on the right in the Figure 4.20, controls the look-and-feel capabilities of each Swing component by communicating with the component's UI object, as shown in the diagram.

The UI Delegate: In the Swing API, the name of each generic component class is preceded by a "J" -- for example, classes in the Swing set are named JButton, JTree, JTable, and so on. Each generic component class handles its own individual own view-and-controller responsibilities. But, as shown in the preceding diagram, each class delegates the look-and-feel-specific aspects of its responsibilities to whatever UI object the currently installed look-and-feel provides. For this reason, the UI object that's built into every Swing component is sometimes referred to as the UI *delegate*.

The Component UI Class: In Swing, all UI delegate classes are derived from a superclass named Component UI. This class contains most of the important machinery for making Swing's pluggable look-and-feel work. Its methods deal with UI installation and uninstallation, and with delegation of a component's geometry-handling and painting. Delegate UIs are described in more detail later, in the Pluggable Look and Feel section of this article.

The Component Model Interface: In component design circles, it's generally considered good practice to center the architecture of an application around its data rather than around its user interface. To support this paradigm, Swing defines a separate model interface for each kind of component that can store or manipulate values or data. This separation provides Swing programs with the option of plugging in their own customized model implementations. You can use the Default or a Custom Component Models.

If an application doesn't explicitly provide its own model implementation for a particular component, Swing creates and installs a default model implementation which implements that component's model interface.

4.2.1.3.2. Overview of the Swing Components

A list of the Swing components is contained in the Appendix E.

The main categories are:

1.Basic Controls : Exist primarily to get input from the user; generally also show simple state.

- Buttons (JButton, JRadioButton, JCheckBox, JToggleButton)
- Combo box (JComboBox)
- List (JList)
- Menu (JMenu, JMenuBar, JMenuItem)
- Slider (JSlider)
- ScrollBar (JScrollBar)
- Text fields (JTextField)
- Tool bar (JToolBar)

2. *Editable Displays of Formatted Information*: Display highly formatted information that can be edited by the user.
   - Table (JTable)
   - Text (JTextField, JPasswordField , JTextArea)
   - Tree (JTree)
   - Color chooser (JColorChooser)
   - File chooser (JFileChooser)

3. *Uneditable Information Displays*: Exist solely to give the user information.
   - Label (JLabel, ImageIcon)
   - Progress bar (JProgressBar)
   - Tool tip (JToolTip)

4. *Space-Saving Containers*: Display more information in less space.
   - Scroll pane (JScrollPane)
   - Split pane (JSplitPane)
   - Tabbed pane (JTabbedPane)

4.2.1.4. Conclusion

Our High Level Components Library will be designed in Java, using the Swing architecture and Java Beans conventions.

We decided to create our own L&F which we will call the "Hyper-Metal" L&F. This new L&F is an alteration of the Metal L&F. The reason for this is that it would have taken too much time to implement three L&F for each components. This means that our components won't be able to switch between the three standard Swing L&F (Windows, Metal and Motif).

4.2.2. Components Design

4.2.2.1. High Level CIO's

Actually most of the AIOs handled by the selection rules present lots of similarities with other AIOs. For instance, the only difference between a List Box (see Figure 3.9) and a Scrollable List Box (see Figure 3.10) is the fact that two "Double Arrows" buttons are added to the scrollbar of the Scrollable List Box. Therefore, one High Level CIO could implement both AIO's. A particular property of this CIO will allow the user to transform the List Box into a Scrollable List Box.

At this point we grouped the 92 AIOs used in the trees (see Table D.1 in Appendix D) according to their similarities. We defined 42 groups that will represent 42 High Level CIO's or components. For example, see the High Level Component List Box and the corresponding AIOs in Table 4.4 below.

High-level Components	AIO's
List Box	List Box
	Multiple Selection List Box
	Graphical List Box
	Scrollable Graphical List Box
	Scrollable List Box
	Scrollable Multiple Selection List Box
	Multiple Selection Graphical List Box
	Scrollable Multiple Selection Graphical List Box

Table 4.4: List Box and the corresponding AIOs

The list of the High Level Components is in the Appendix D.

4.2.2.2. Conventions

In the implementation process of the library we followed the following conventions:

4.2.2.2.1. The LabeledComponent class

Most of the UI components extend the Labeled Component class. The labeled component is the combination of a label with a panel in which it is possible to add components. (see Figure 4.21 and 4.22 below).

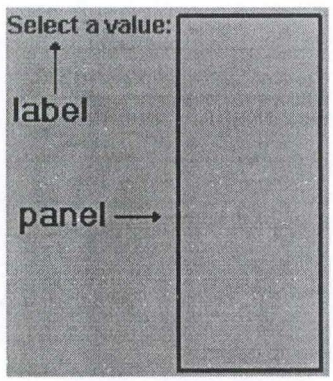


Figure 4.21: A labeled component with the label on the left of the panel.

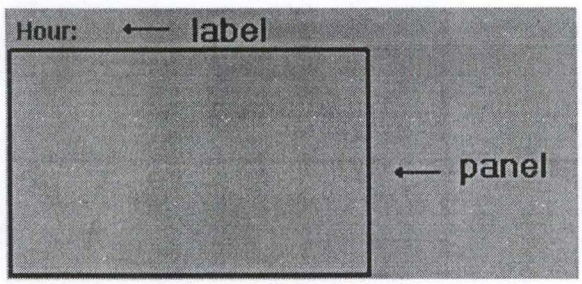


Figure 4.22: A labeled component with the label above the panel.

That means that almost all the UI components of our library contain a label as well as methods to set the position and the alignment of the label.

4.2.2.2.2. The GroupBox class

The Group Box component (see Figure 3.16) is a panel with an optional border in which the user can add components. This component is very useful if the designer wants that several UI components move simultaneously. Moreover, a simplified layout manager is implemented inside the Group Box allowing the user to add a component at a specific position inside the panel.

4.2.2.2.3. Values and selectedValues properties

We decided to follow the following name conventions:

- When the user has to select a value among a list or a set of possible values, the *value* property of the UI component represents the possible options in which the user can select a value. In that case, the *selectedValue* or *selectedValues* property represents the value(s) selected by the user among the possible values. For instance, the setValue method of the List Box (see Figure 3.9) sets the content of the list while the setSelectedValue method sets which ones of the values displayed in the list are selected or not.
- When the user has to select one value and if no possible values are presented into a particular component, the *value* property represents the value that he has typed in or selected. For instance, the setValue method of an Edit Box (see Figure 3.5) sets the text of the text field.

4.2.2.2.4. Events fired

Most of the UI components of the library fire events when the value of a key property has changed. The key properties are most of the time the "values" or "selectedValues" properties (see 4.2.2.2.3 above) . If the user wants to listen for a property change event, he needs to add a Property Change Listener to the component as shown in Figure 4.23.

```
myComponent.addPropertyChangeListener(new PropertyChangeListener ()
{
        public void propertyChange(PropertyChangeEvent e)
        {
                if (e.getPropertyName() == "value")
                {
                // code if the "value" property has changed
                }

        }
});
```

Figure 4.23: How to listen for a change of the "value" property.

4.2.2.2.5. Hyper-Metal Look and Feel

We tried to develop UI components that look the same as the Swing components in their Metal Look and Feel (see Appendix E). For instance, the look of the Switch component uses the color as well as the texture of the Swing JScrollBar component as shown in the Figure 4.24 below.

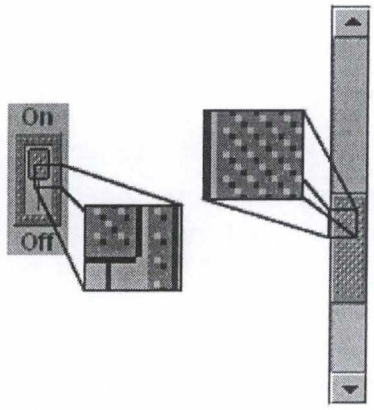


Figure 4.24: A Switch compared with a Swing JScrollBar.

4.2.2.2.6. Editable and Non-editable colors.

We decided to define two different colors that will be used each time that a component is *editable* or *non-editable*. For instance, the Combo Box (see Figure 4.25 below) is the combination of an Edit Box, which is editable, with a List Box, which is non-editable.

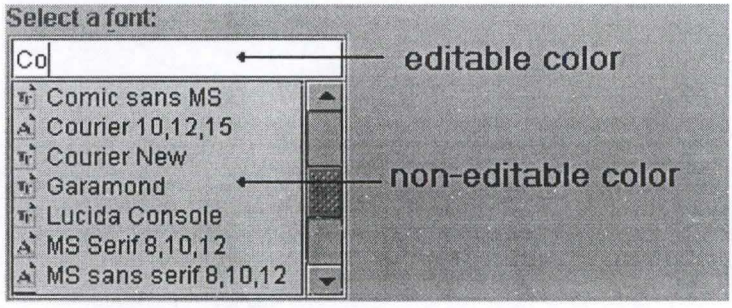


Figure 4.25: How we used the Editable and Non-editable colors.

That means that the user will be able to make the difference quite easily between two UI components whose only difference is the fact that they are editable or not. For instance, the Figure 4.26 below shows a Unitary List Box and a Spin Button. The only visible difference is the color of the text field. The color of the text field of the Spin Button is white (= editable color) while the color of the text field of the Unitary List Box is the non-editable color.

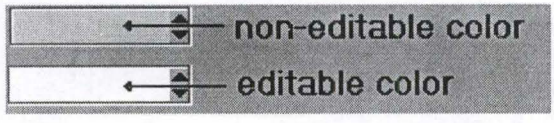


Figure 4.26: A Spin Button and an Unitary List Box

4.2.2.2.7. Constants

Most of the constants needed to customize the UI components are stored into the Constants class.

For instance, if the user wants to change the orientation of a Switch (see Figure 3.40), he will write:

```
MySwitch.setOrientation(Constants.VERTICAL);
```

4.2.2.2.8. ToolBox

All the pictures, borders, internal constants as well as some very useful private methods are stored into the ToolBox class. This class' access is private.

4.2.2.2.9. Orientation

We tried as often as possible to create two different orientations for each UI component. The two Figures below show a vertical Date Spinner Accumulator (Figure 4.27) as well as a horizontal Date Spinner Accumulator (Figure 4.28).

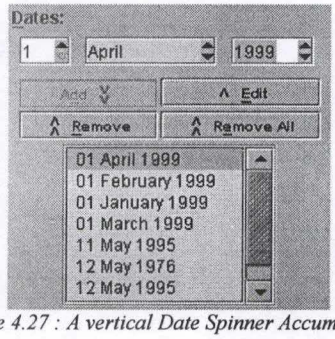


Figure 4.27 : A vertical Date Spinner Accumulator

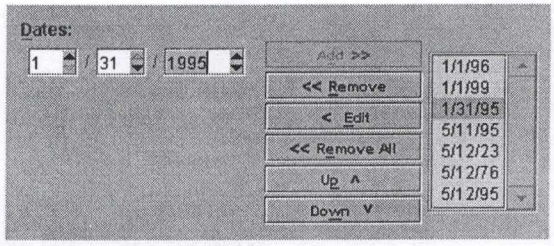


Figure 4.28: A horizontal Date Spinner Accumulator

4.2.2.3. Application Programming Interface (API)

The description of all the components can be found in the API. You can consult the API on our site at : XXX.

5. System evaluation

In order to evaluate our work, we decided to create a form using some of our UI components. The goal of this form will be the registration of a student for courses.

5.1. Class Model

The class model for the student registration is shown in the Figure 5.1 below.

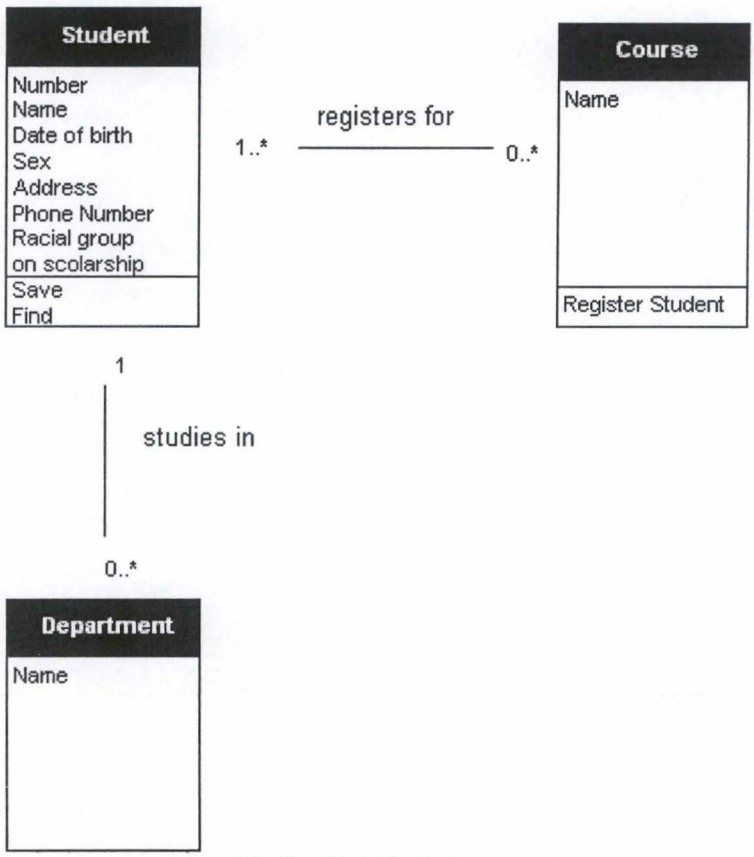


Figure 5.1: Class Model for Student registration

5.2. Selection of AIOs

5.2.1. Student Attributes

1) Number

This attribute represents the registration number of the student.

Path followed in the selection tree:
- The value of the interaction type criterion is Input
- The type of the value is Integer.
- The number of values to choose is 1
- The domain is unknown

The AIO selected is a Spin Button as shown in the Figure 4.16.

2) Name

This attribute represents the full name of the student.

Path followed in the selection tree:
- The value of the interaction type criterion is Input
- The type of the value is Alphanumeric
- The number of values to choose is 1
- The domain is unknown
- The length of the name is inferior to 40 (see Lm in Table 4.1)

As shown in the Figure 5.2 below, the selected AIO is a Single-Line Edit Box.

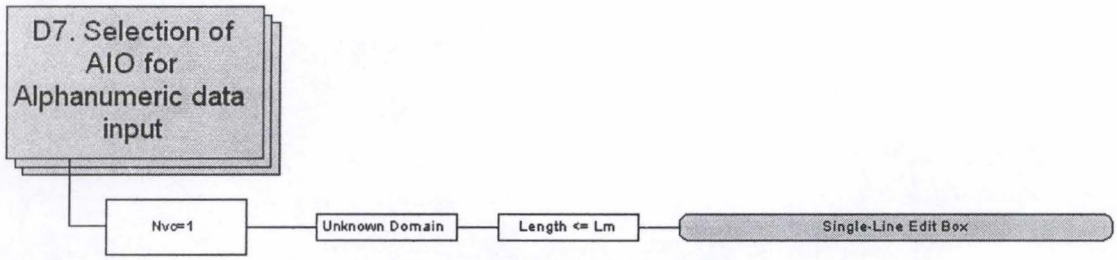


Figure 5.2: The selection path of the AIO for the Name attribute.

3) Date of birth

This attribute represents the date of birth of the student.

Path followed in the selection tree:
- The value of the interaction type criterion is Input
- The type of the value is Date
- The number of values to choose is 1
- The domain is unknown
- The preference for selection is true

As shown in the Figure 5.3 below, the selected AIO is a Date Spinner.

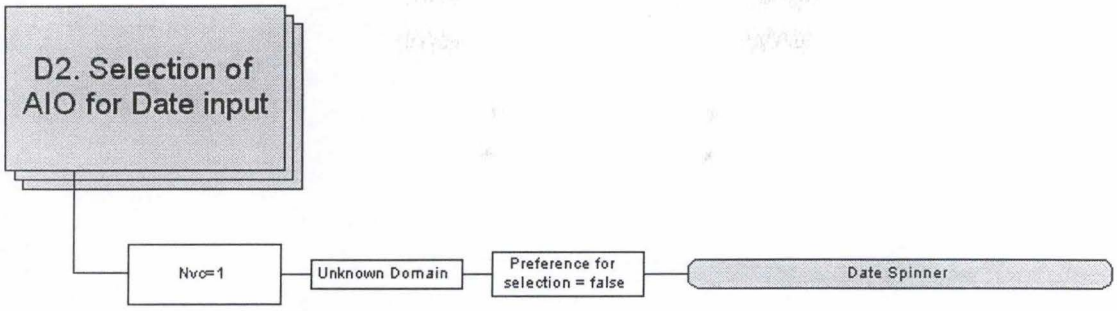


Figure 5.3: The selection path of the AIO for the Date of birth attribute.

4) Sex

This attribute represents the sex of the student.

Path followed in the selection tree:
- The value of the interaction type criterion is Input
- The type of the value is Boolean because the number of possible values for this attribute is 2 (Male or Female)
- The two possible values are opposite values.

As shown in the Figure 5.4 below, the selected AIO is a Switch.

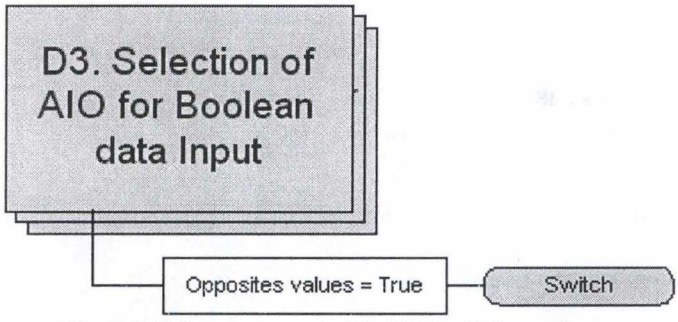


Figure 5.4: The selection path of the AIO for the Sex attribute.

5) Address.

This attribute represents the address of the student.

Path followed in the selection tree:
- The value of the interaction type criterion is Input
- The type of the value is Alphanumeric
- The number of values to choose is 1
- The domain is unknown
- The length of the name is superior to 40 (see Lm in Table 4.1)

As shown in the Figure 5.5 below, the selected AIO is a Multi-Line Edit Box

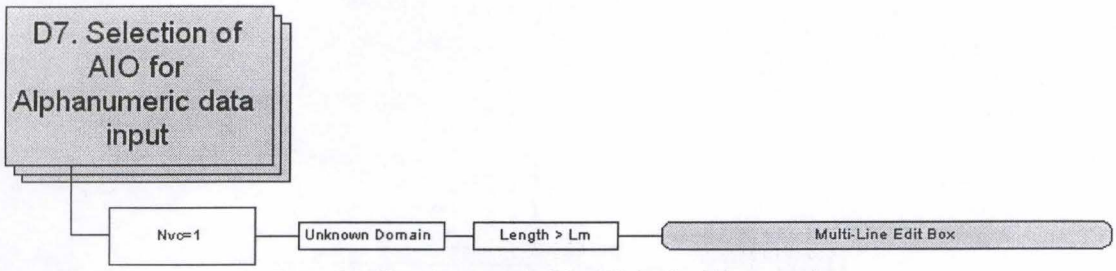


Figure 5.5: The selection path of the AIO for the Address attribute

6) Phone Number

This attribute represents the phone number of the student.

The selection is the same as for the name attribute, and therefore the selected AIO is a Single-Line Edit Box.

7) Population group

This attribute represents the population group of the student.

Path followed in the selection tree:

- The value of the interaction type criterion is Input
- The type of the value is Alphanumeric
- The number of values to choose is 1
- The domain is mixed because in addition to the usual values (Black, White, Coloured, Asian), some other values are possible.
- The number of possible values is between 4 and 7 (= Magical Number, see Table 4.1).

As shown in the Figure 5.6 below, the selected AIO is a Radio Button + Label + Edit Box + Group Box.

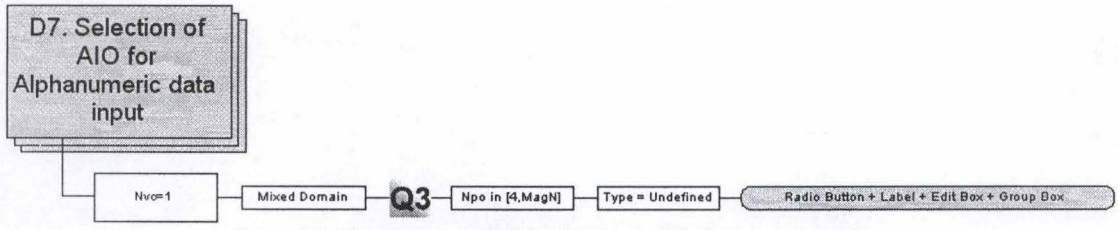


Figure 5.6: The selection path of the AIO for the Population group attribute

8) On scholarship

This attribute defines whether or not the student received a scholarship.

Path followed in the selection tree:

- The value of the interaction type criterion is Input

- The type of the value is Boolean because the number of possible values for this attribute is 2 (true or false)
- The two possible values are not opposite values.

As shown in the Figure 5.7 below, the selected AIO is a Check Box.

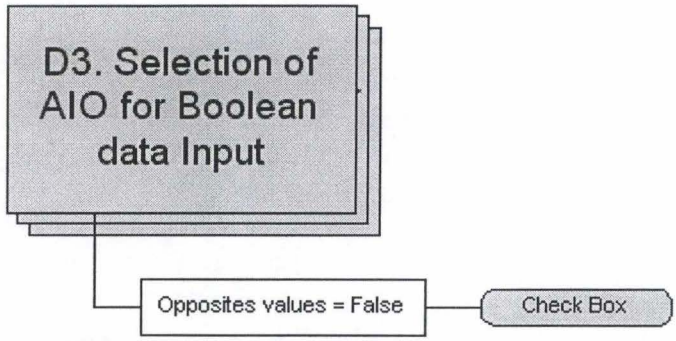


Figure 5.7: The selection path of the AIO for the on scholarship attribute

5.2.2. Department Attribute

The only attribute of a department is the name. A student studies in one and only one department.

Path followed in the selection tree:
- The value of the interaction type criterion is Input
- The type of the value is Alphanumeric
- The number of values to choose is 1
- The domain is known
- The number of possible values is between 7 (Magical number) and 50 (Tm, see Table 4.1)

As shown in the Figure 5.8 below, the selected AIO is a List Box.

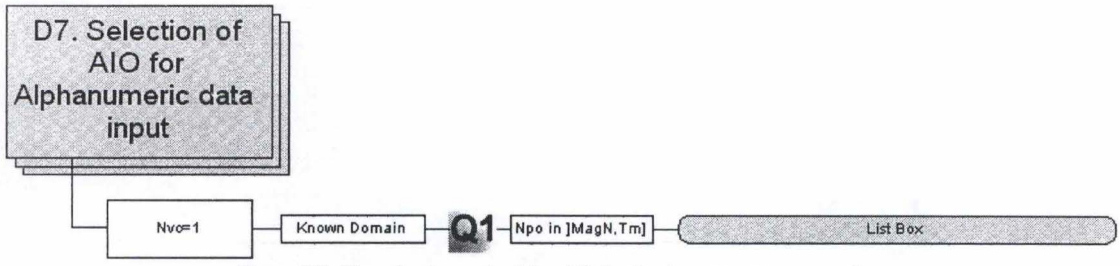


Figure 5.8: The selection path of the AIO for the department name attribute

5.2.3. Courses Attribute

The only attribute of a course is its name. The student can register for one or more than one different courses.

Path followed in the selection tree:

- The value of the interaction type criterion is "Input"
- The type of the value is Alphanumeric
- The number of values to choose is superior to 1.
- The domain is known
- The number of possible values is over 50 (Tm, see Table 4.1).

As shown in the Figure 5.9 below, the selected AIO is a Scrollable Non-editable Accumulator.

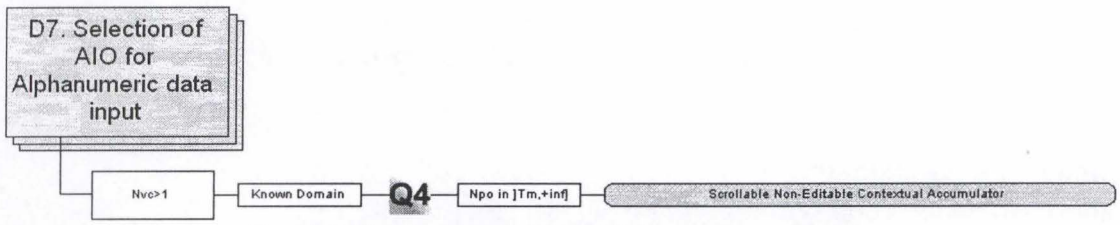


Figure5.9 : The selection path of the AIO for the courses name attribute

5.2.4. AIOs recapitulation

Attribute name	**AIO**
Number	Spin Button
Name	Edit Box
Date of birth	Date Spinner
Sex	Switch
Address	Multi-Line Edit Box
Phone number	Edit Box
Population group	Radio Button + Label + Edit Box + Group Box
On scholarship	Check Box
Department Name	List Box
Courses names	Scrollable Non-editable Accumulator

Table 5.1: Summary of the attributes and the selected AIOs

5.3. Alternatives

At this stage, we must consider for each AIO whether or not it is pertinent to replace it with its low-density alternative. The possible alternatives are in the Table C.1 (Appendix C).

Let's decide for instance to replace the List Box selected for the department name attribute with a Drop-Down List Box.

5.4. CIOs

For each AIO, we must now define which component of our library will be used. The Table D.1 (Appendix D) shows for each AIO which High Level Component of our library should be used.

The Table 5.2 below shows for each attribute, the component of our library that will be used.

Attribute name	**AIO**	**Component of our library**
Number	Spin Button	Spin Button
Name	Edit Box	Edit Box
Date of birth	Date Spinner	Date Spinner
Sex	Switch	Switch
Address	Multi-Line Edit Box	Multi-Line Edit Box
Phone number	Edit Box	Edit Box
Population group	Radio Button + Label + Edit Box + Group Box	Text Radio Group
On scholarship	Check Box	Check Box
Department Name	Drop Down List Box	Drop Down List Box
Courses names	Scrollable Non-editable Accumulator	List Box Accumulator

Table 5.2: Matching between the AIOs and the components of our library

The "Find", "Save" and "Register" methods will be implemented with the use of the Swing JButton component.

5.5. Implementation

The Figure 5.10 below shows a picture of the form generated.

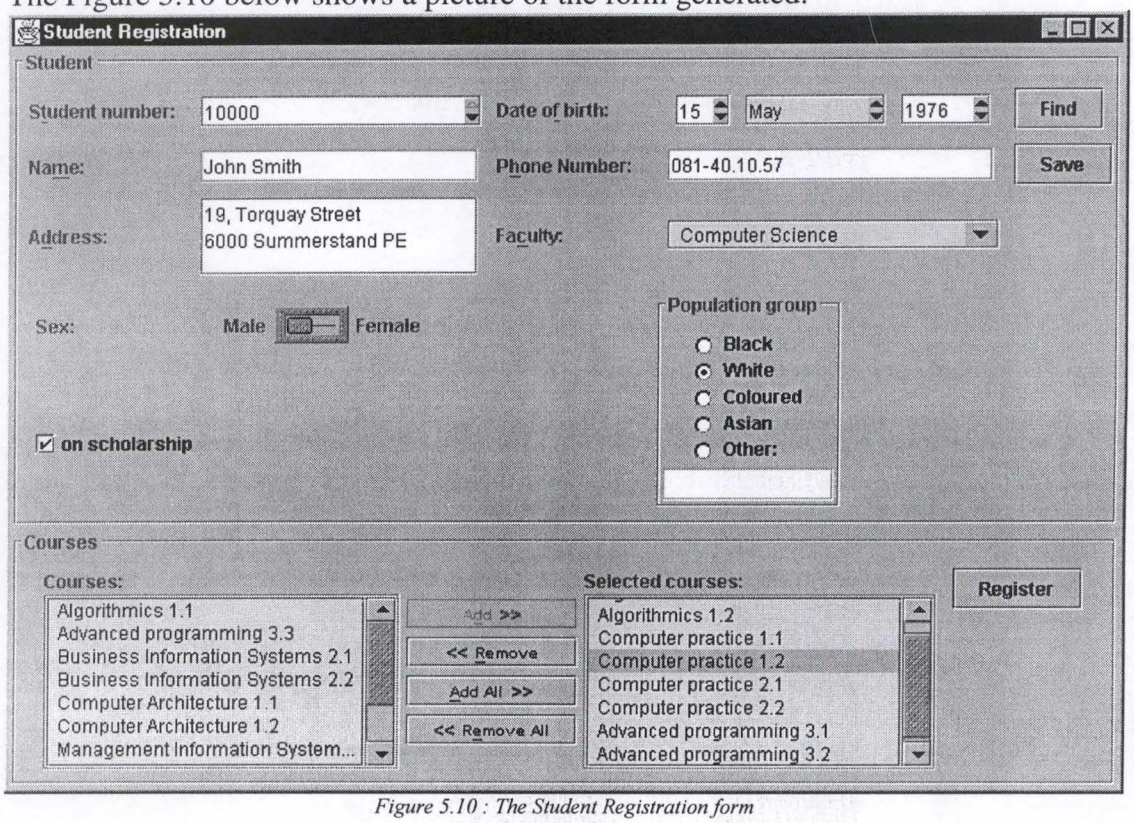


Figure 5.10 : The Student Registration form

The code written in order to implement this form is the following:

```
public class TestUPE
{

public static void main(String[] args)
{
/* GroupBox containing the information about the student*/
GroupBox groupStudent = new GroupBox();
GroupStudent.setTitle("Student");
/* Student Number */
SpinButton number = new SpinButton();
number.setLabel("Student number");
number.setMnemonic('t');
number.setMaximum(10000);
number.setColumns(15);
number.setComponentHorizontalLocation(110);
groupStudent.addComponent(number,1,1,1,1,Constants.LEFT);
/* Student Name */
EditBox name = new EditBox();
name.setLabel("Name");
name.setMnemonic('m');
```

```
name.setColumns(16);
name.setComponentHorizontalLocation(110);
groupStudent.addComponent(name,2,1,1,1,Constants.LEFT);
/* Address */
MultiLineEditBox address = new MultiLineEditBox();
address.setLabel("Address");
address.setMnemonic('d');
address.setEditBoxSize(new                                    Dimen-
sion(name.getComponentSize().width,50));
address.setComponentHorizontalLocation(110);
address.setLabelCentered(false);
groupStudent.addComponent(address,3,1,1,1,Constants.LEFT);
/* Date of birth */
DateSpinner dateOfBirth = new DateSpinner();
dateOfBirth.setLabel("Date of birth");
dateOfBirth.setCountry(Constants.UK);
dateOfBirth.setMnemonic('f');
dateOfBirth.setDateFormat(Constants.LETTER);
dateOfBirth.setComponentHorizontalLocation(110);
groupStudent.addComponent(dateOfBirth,1,2,1,1,Constants.LEFT);
/* Phone Number*/
EditBox phone = new EditBox();
phone.setLabel("Phone Number");
phone.setComponentHorizontalLocation(110);
phone.setColumns(19);
phone.setMnemonic('h');
groupStudent.addComponent(phone,2,2,1,1,Constants.LEFT);
/* Faculty of the student*/
DropDownListBox faculty = new DropDownListBox();
faculty.setLabel("Faculty");
faculty.setComponentHorizontalLocation(110);
faculty.setAutomaticSize(false);
faculty.setMnemonic('c');
faculty.setCellWidth(dateOfBirth.getComponentSize().width - 30);
faculty.setAlphabeticalOrder(Constants.INCREASING);
faculty.setStringValues(new          String[]          {"Computer          Sci-
ence","Biology","Arts","Law","Physics","Mathematics","Medicine","Zoology","B
otanics"});
groupStudent.addComponent(faculty,3,2,1,1,Constants.LEFT);
/* Scholarship */
CheckBox scholarship = new CheckBox();
scholarship.setText("on scholarship");
groupStudent.addComponent(scholarship,5,1,1,1,Constants.LEFT);
/* Sex of the student */
Switch sex = new Switch();
sex.setLabel("Sex");
sex.setOnValue("Male");
sex.setOffValue("Female");
sex.setComponentHorizontalLocation(110);
groupStudent.addComponent(sex,4,1,1,1,Constants.LEFT);
```

```
/* Population group */
TextRadioGroup populationGroup = new TextRadioGroup();
PopulationGroup.setTitle("Population group");
PopulationGroup.setValues(new String[] {"Black","White","Coloured","Asian"});
GroupStudent.addComponent(populationGroup,4,2,2,1,Constants.CENTER);
GroupStudent.setComponentSpacing(5,5);
/* Find button */
JButton find = new JButton("Find");
GroupStudent.addComponent(find,1,3,1,1,Constants.LEFT);
/* Save button */
JButton save = new JButton("Save");
GroupStudent.addComponent(save,2,3,1,1,Constants.LEFT);
/* Group Box containing the information about the courses */
GroupBox groupCourses = new GroupBox();
GroupCourses.setTitle("Courses");
/* courses */
ListBoxAcc courses = new ListBoxAcc();
courses.setCellWidth(200);
courses.setValuesLabel("Courses");
courses.setSelectedValuesLabel("Selected courses");
courses.setListType(Constants.SET);
courses.setStringValues(new String[] {"Algorithmics 1.1","Algorithmics
2.1","Algorithmics 1.2","Algorithmics 2.1","Computer practice 1.1","Computer
practice 1.2","Computer practice 2.1","Computer practice 2.2","Advanced pro-
gramming 3.1","Advanced programming 3.2","Advanced programming
3.3","Business Information Systems 2.1","Business Information Systems
2.2","Computer Architecture 1.1","Computer Architecture 1.2","Management
Information Systems 3.1","Management Information Systems 3.2","End User
Computing 1.1","End User Computing 1.2"});
groupCourses.addComponent(courses,1,1,4,4,Constants.LEFT);
/* Button */
JButton register = new JButton("Register");
GroupCourses.addComponent(register,1,5,1,1,Constants.RIGHT);
GroupCourses.setComponentSpacing(5,5);
/* Frame */
JFrame frame = new JFrame("Student Registration");
frame.getContentPane().add("North",groupStudent);
frame.getContentPane().add("South",groupCourses);
WindowListener l = new WindowAdapter() {
    public void windowClosing(WindowEvent e) {System.exit(0);}    };
frame.addWindowListener(l);
frame.pack();
frame.show();
}
}
```

6. Conclusion

Here is the conclusion that we came to:

- The selection trees handle rules that can meet the requirements of most users. The main existing data types are handled (Time, Date, Graphical,...) as well as input and display components.
- The library of components contains more than 40 High Level Components, that is to say more than most of the IDE available on the market, and covers the main needs in the building of user interfaces. Some of the components are not working perfectly at the moment and will be considerably improved in the next months.

7. References

Nichols, L. (1998). *A Visual Environment for Reusable Object Design* At http://www.cs.upe.ac.za/staff/csalen/PHD1.htm

Vanderdonckt, J. (1993) *A Corpus on Selection Rules for Choosing Interaction Objects.* Technical Report, number 93/3, FUNDP Namur Institute of Computer Science.

Vanderdonckt, J. (1995) *Knowledge-Based Systems for Automated User Interface Generation : the TRIDENT Experience.* Technical Report, In Proceedings of CHI'95 – May 7-8 1995, Denver, Colorado, USA, 21-33.

Vanderdonckt, J. (1997) *Conception assistee de la presentation d'une interface homme-machine ergonomique pour une application de gestion hautement interactive.* PHD Thesis . Insitut d'Informatique, Namur, Belgium

Sun Microsystems (1998,1), *The Java Tutorial* at http://java.sun.com/docs/books/tutorial/

Sun Microsystems (1998,1) *The Swing Connection* at http://java.sun.com/products/jfc/tsc/

8. Appendix

8.1. Vanderdonckt's selection trees

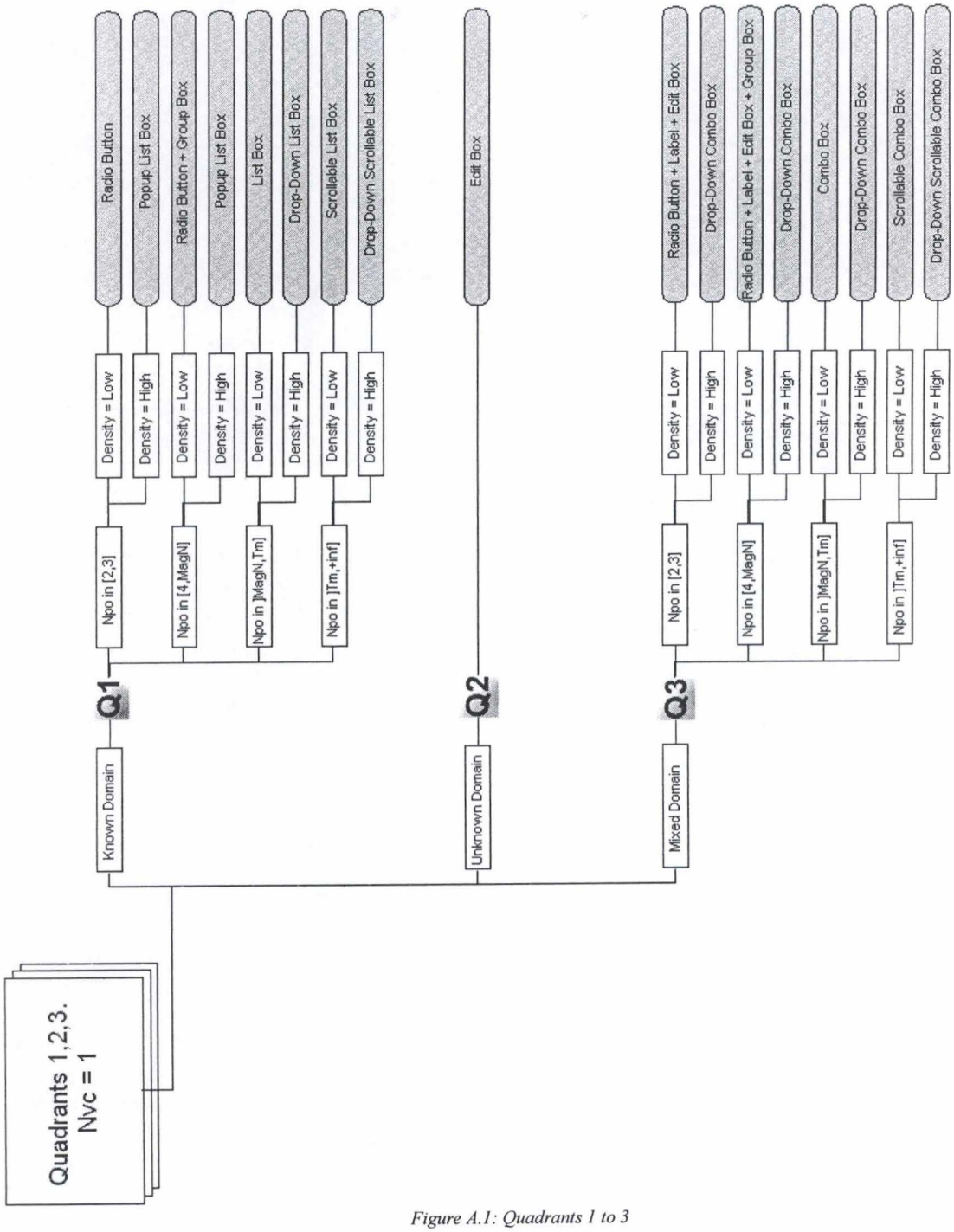


Figure A.1: Quadrants 1 to 3

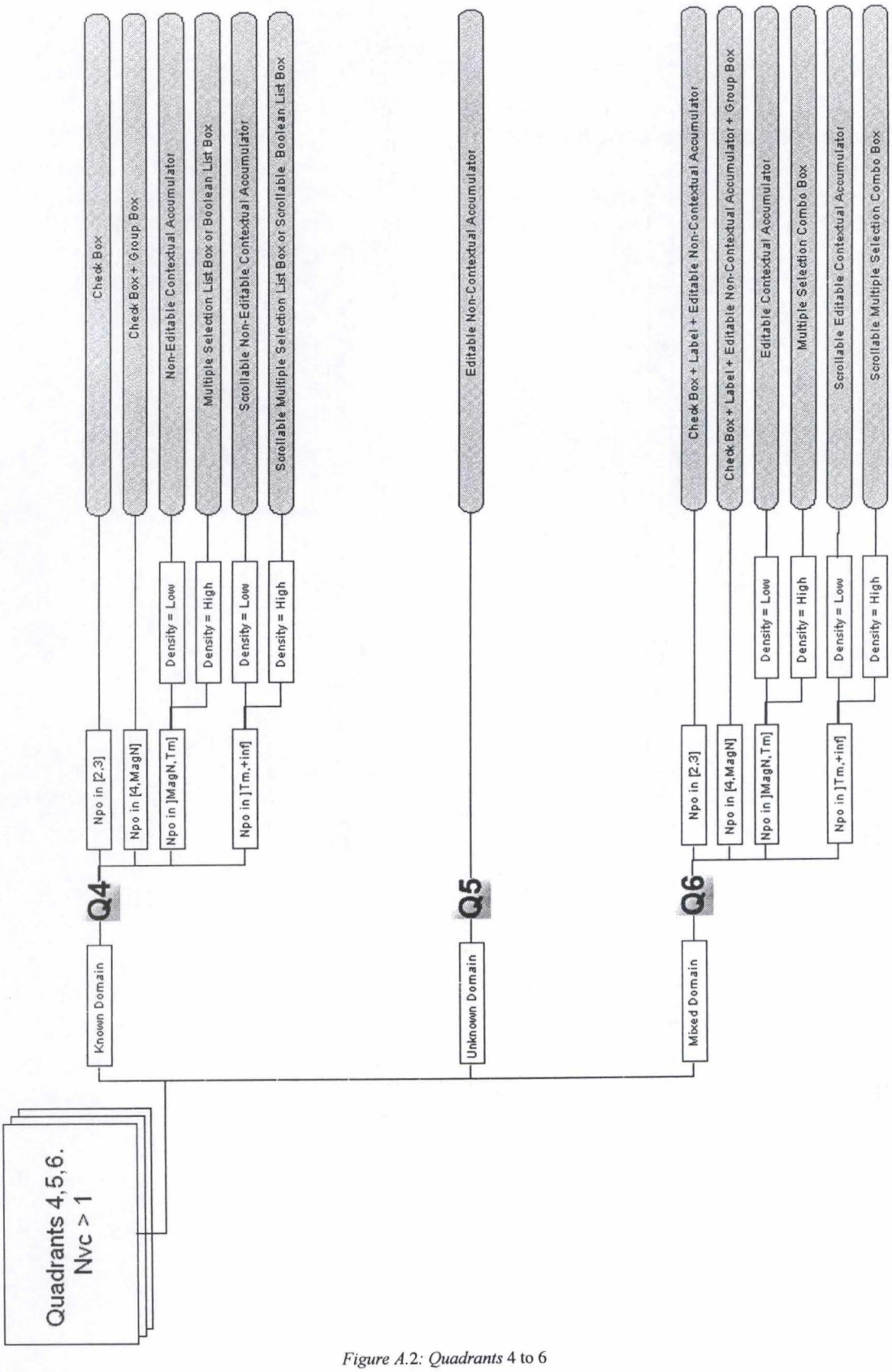


Figure A.2: Quadrants 4 to 6

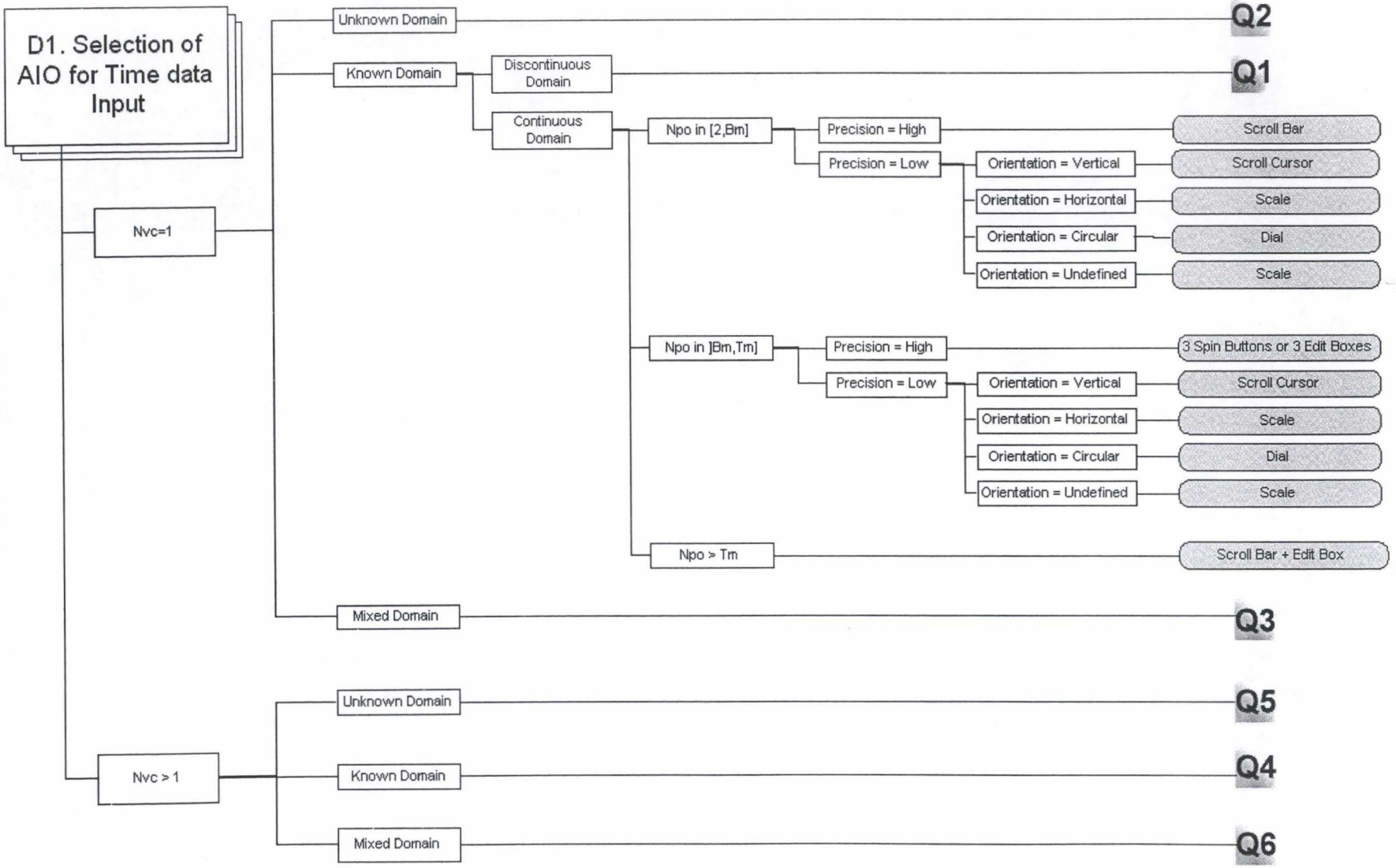


Figure A.3: the Time input selection tree

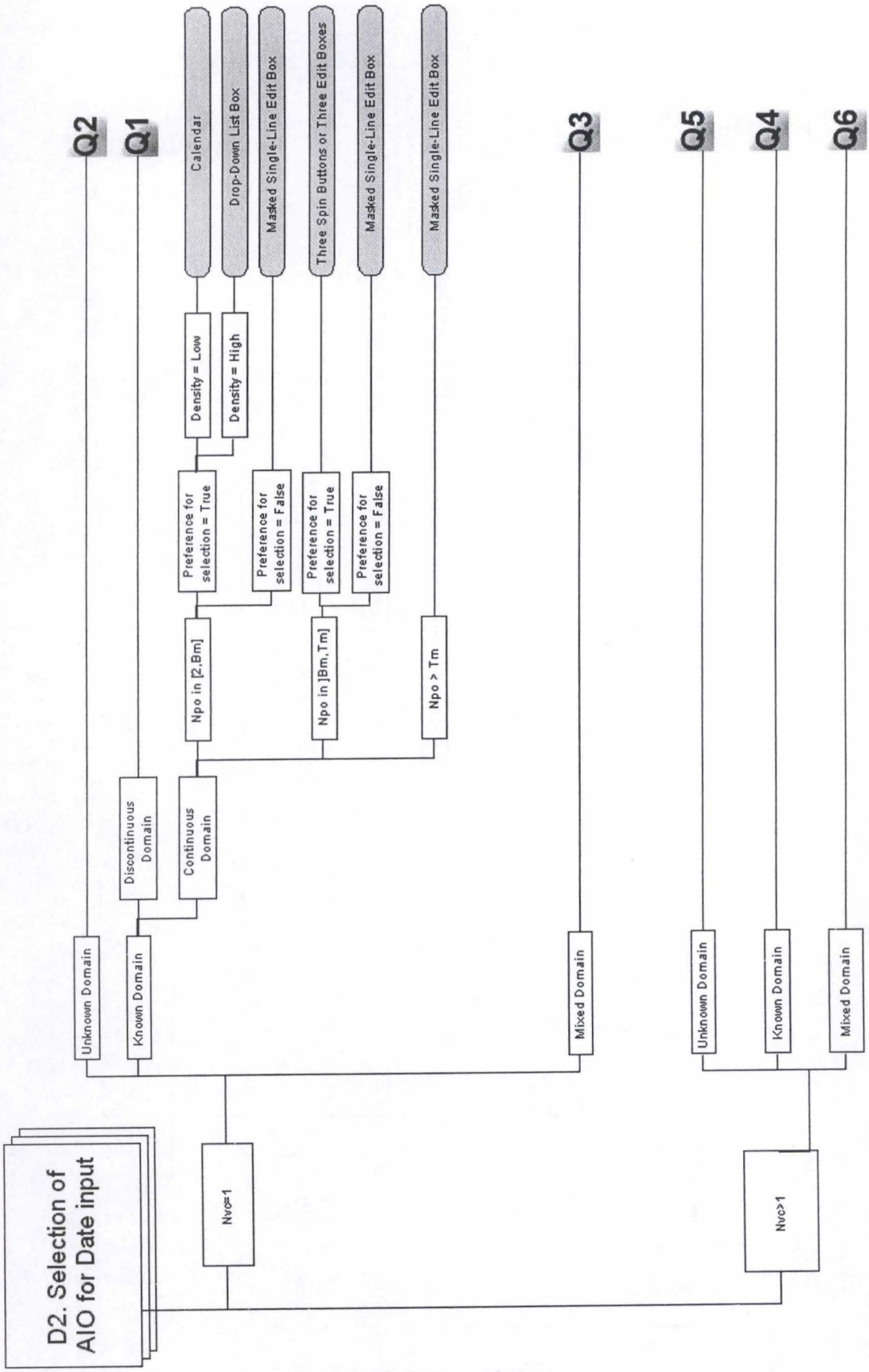


Figure A.4: the Date input selection tree

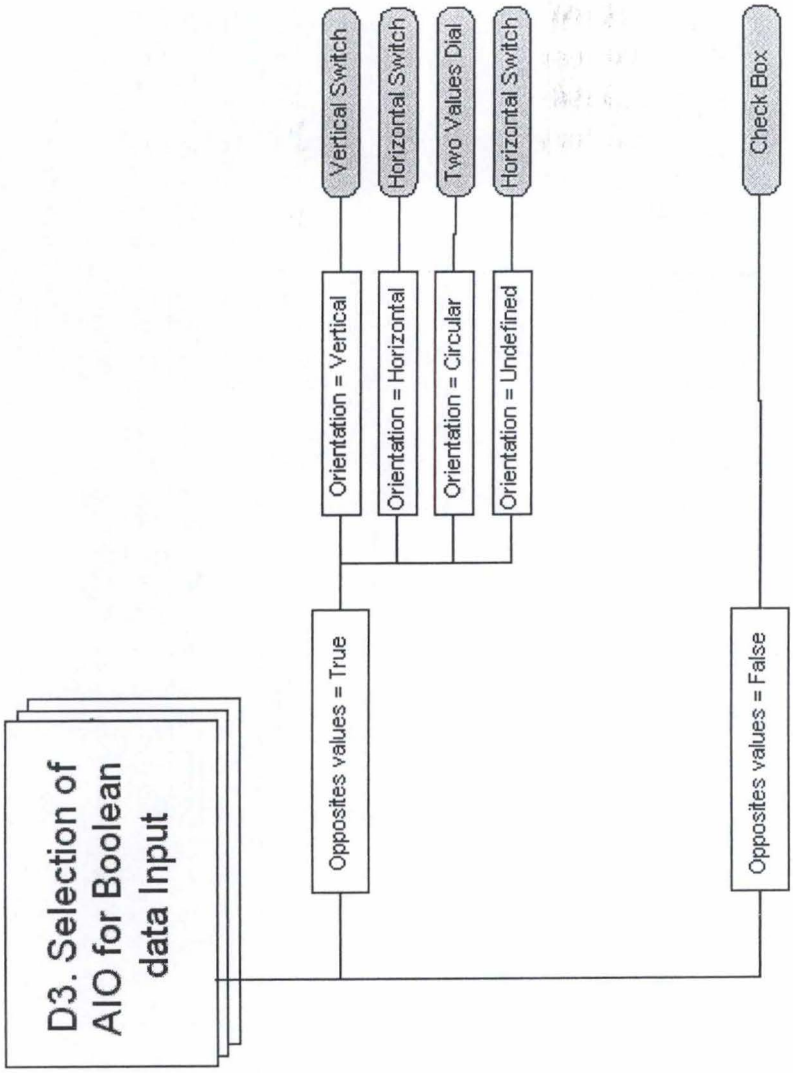


Figure A.5: the Boolean input selection tree

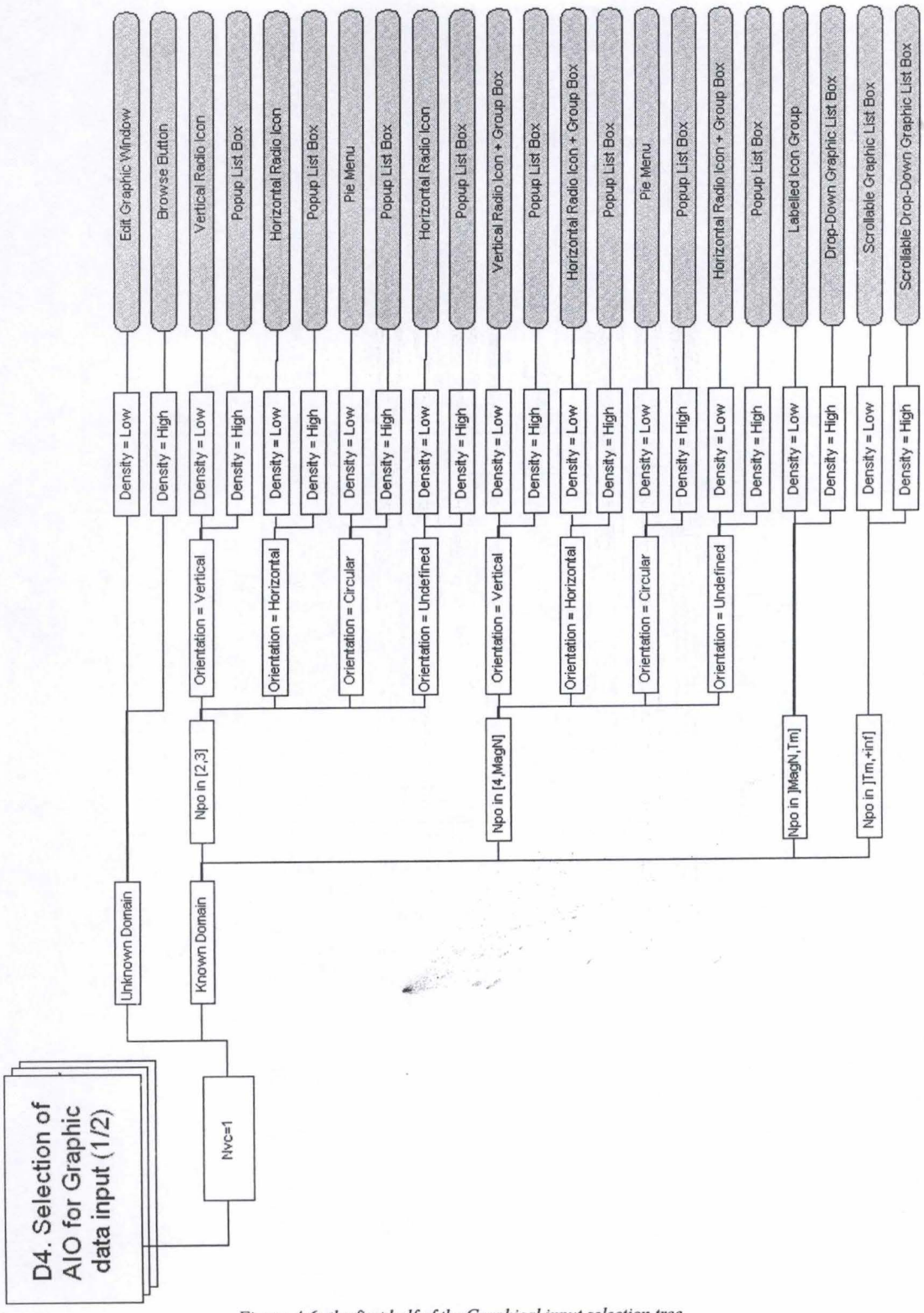


Figure A.6: the first half of the Graphical input selection tree

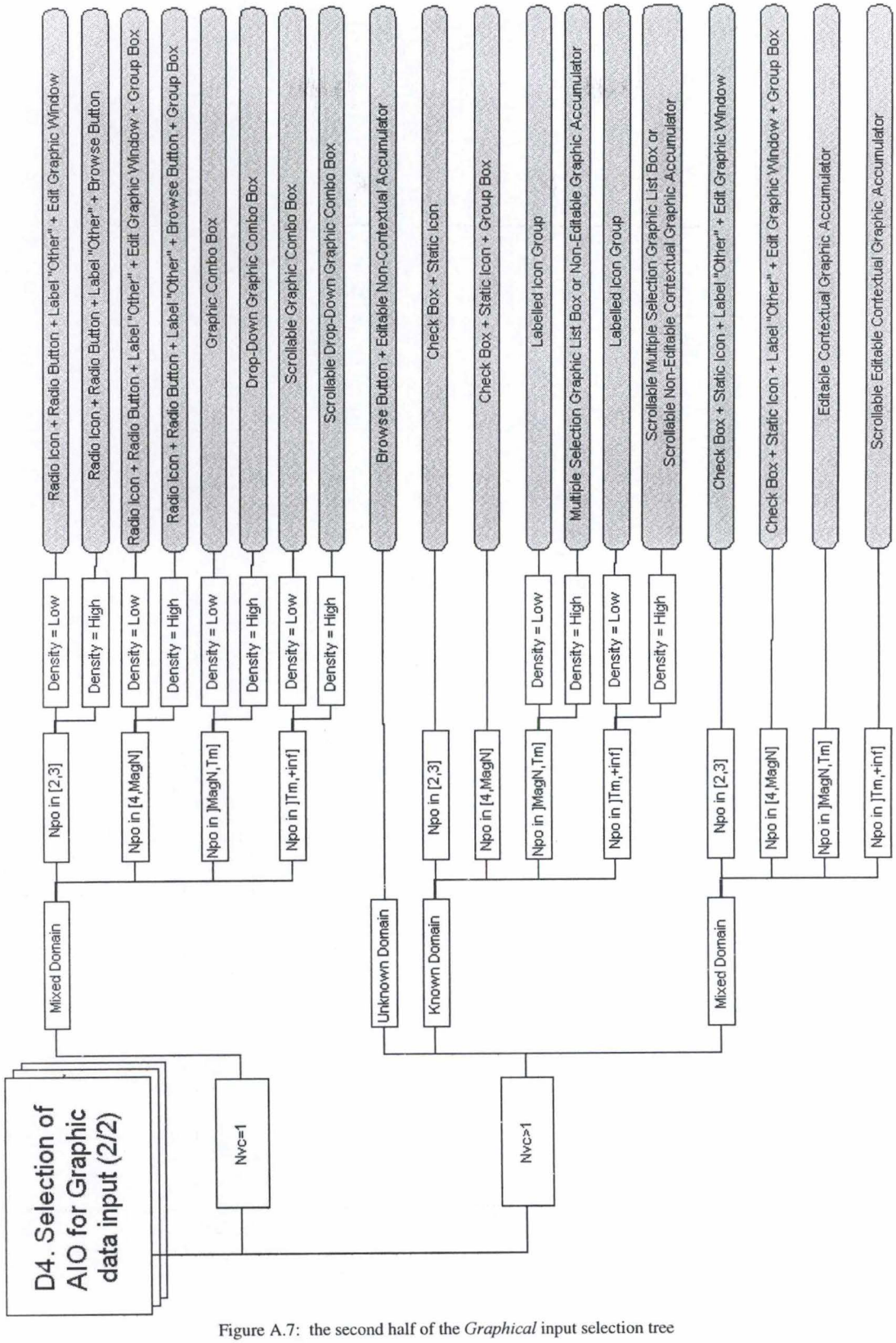


Figure A.7: the second half of the *Graphical* input selection tree

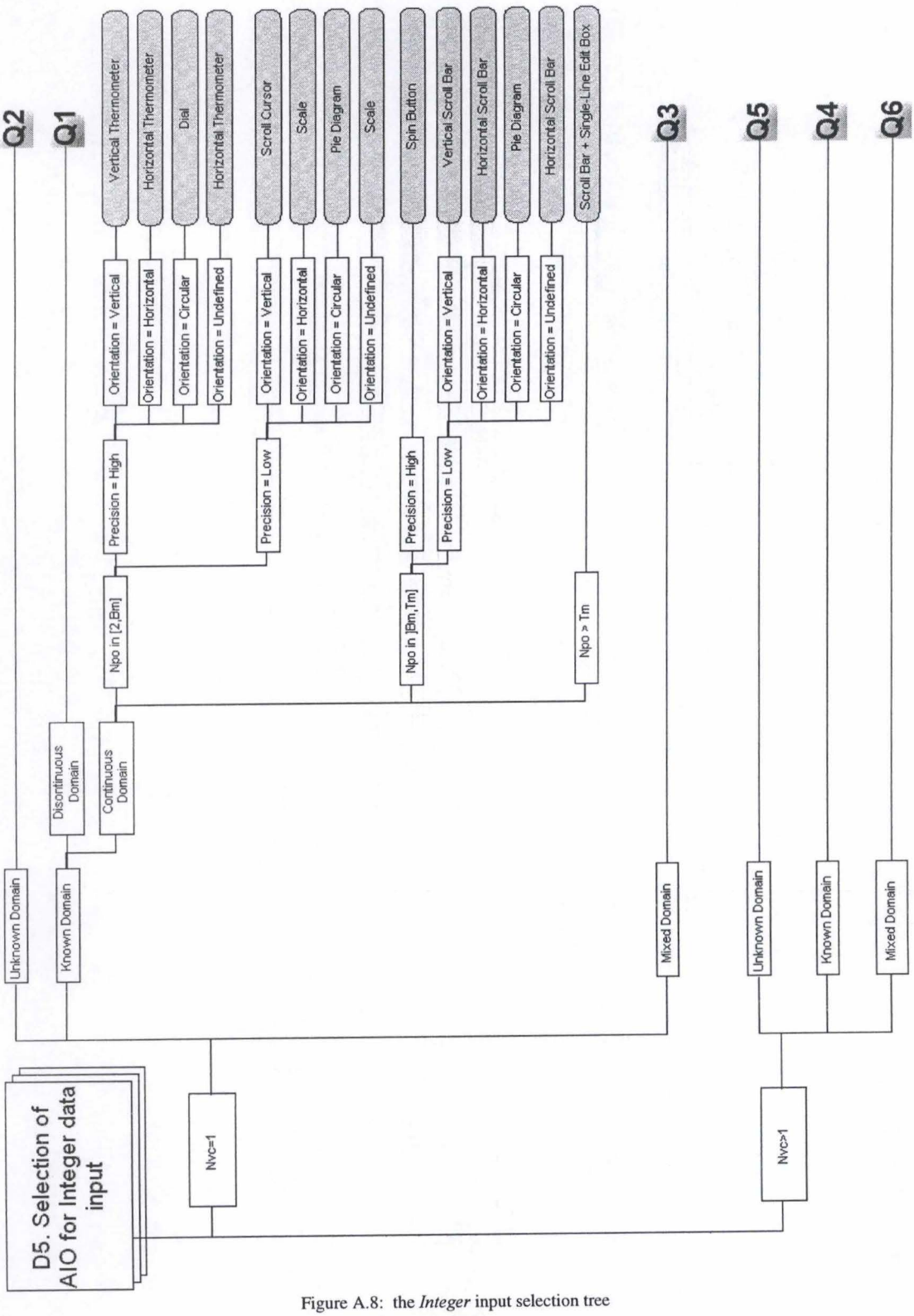


Figure A.8: the *Integer* input selection tree

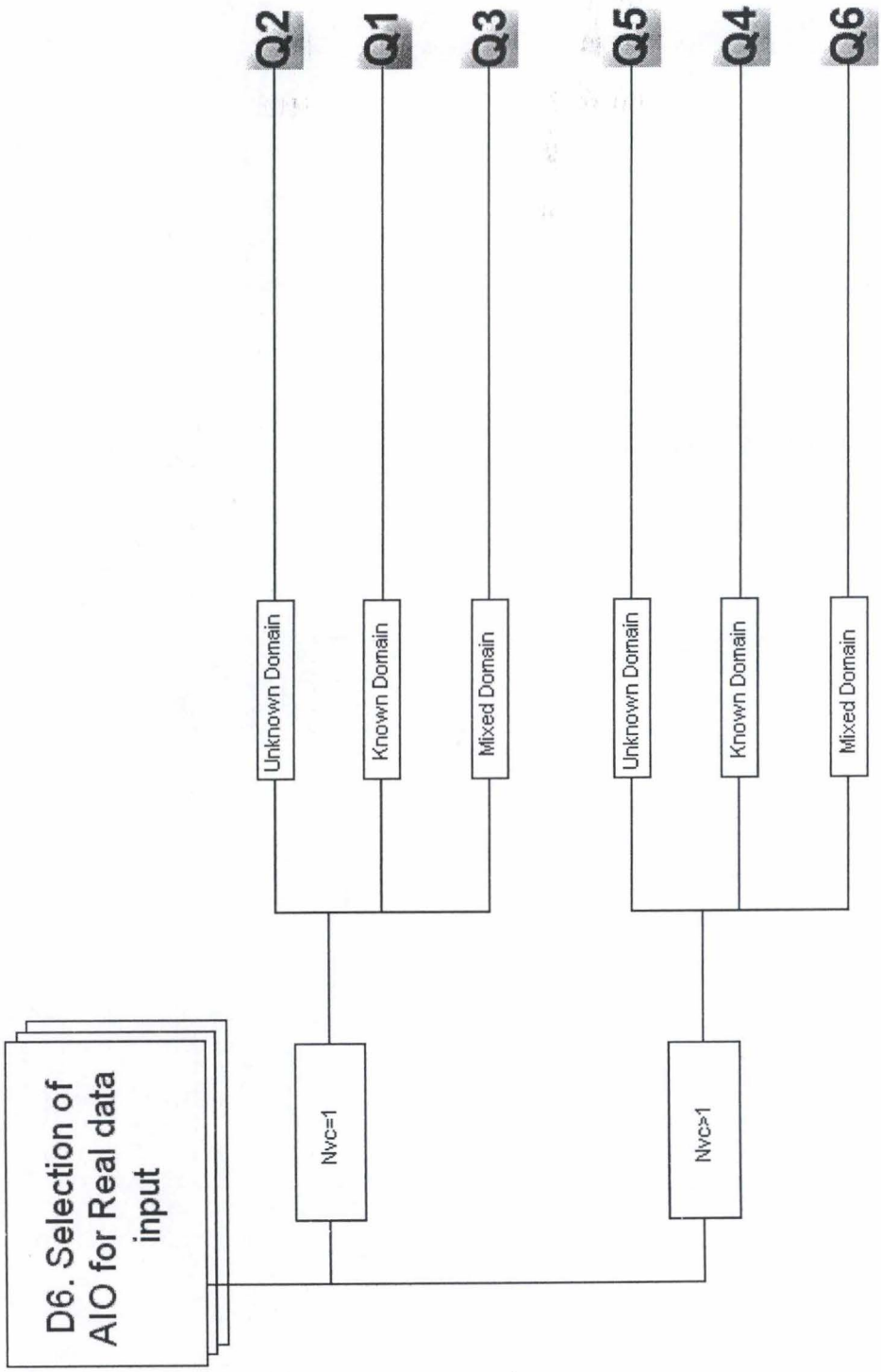


Figure A.9: the Real input *selection* tree

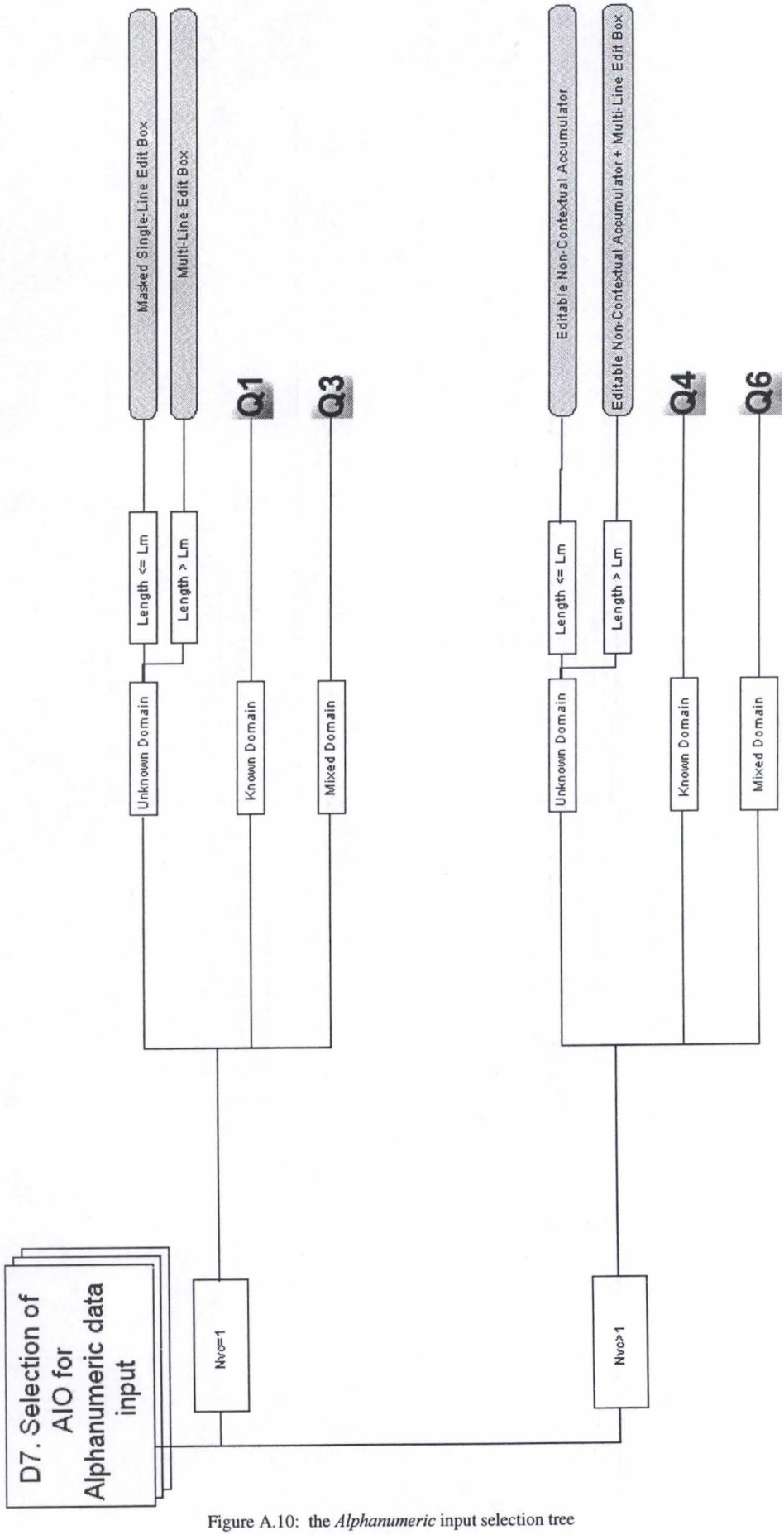


Figure A.10: the *Alphanumeric* input selection tree

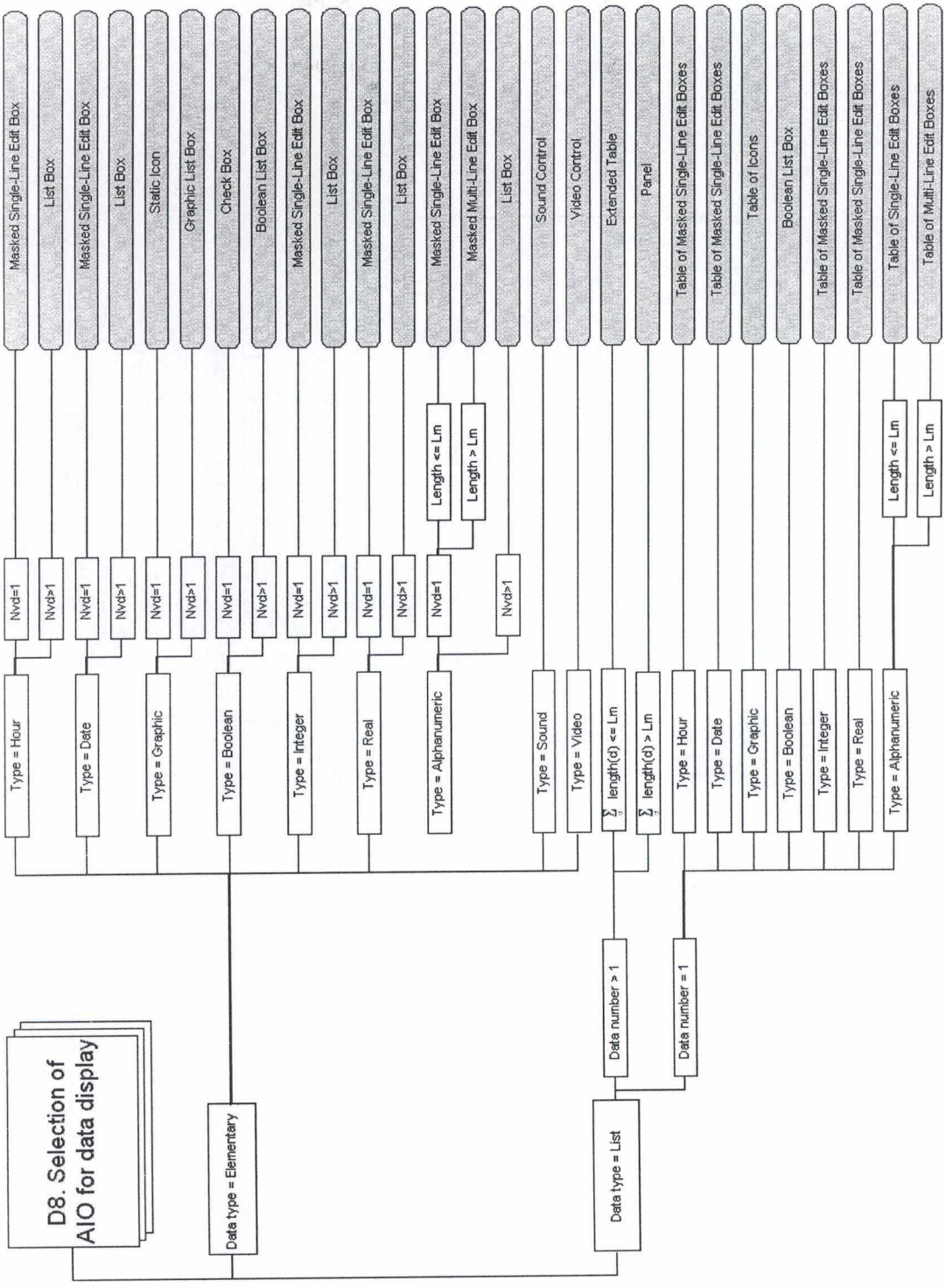


Figure A.11 : The Display Selection tree

8.2. Simplified selection trees

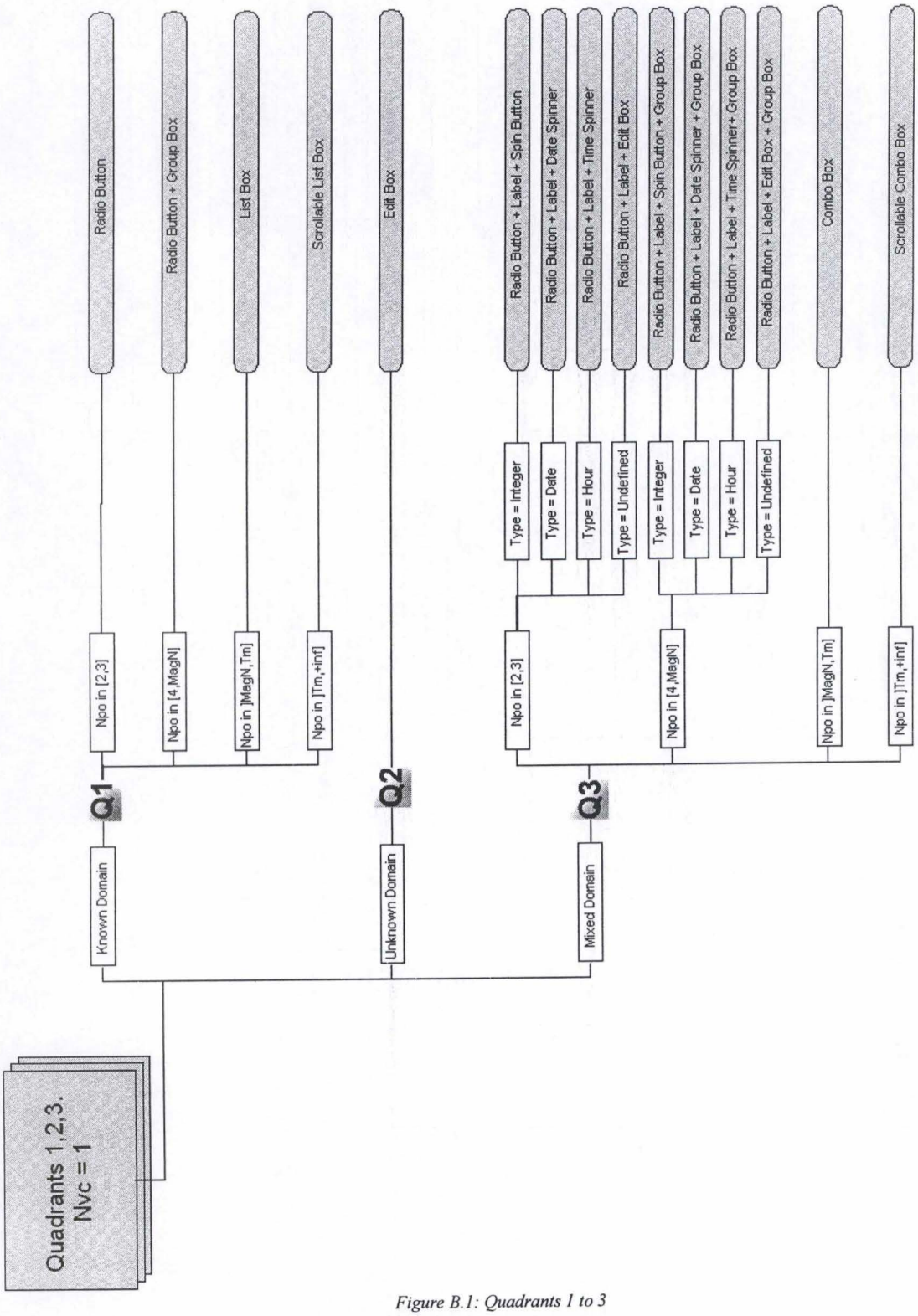


Figure B.1: Quadrants 1 to 3

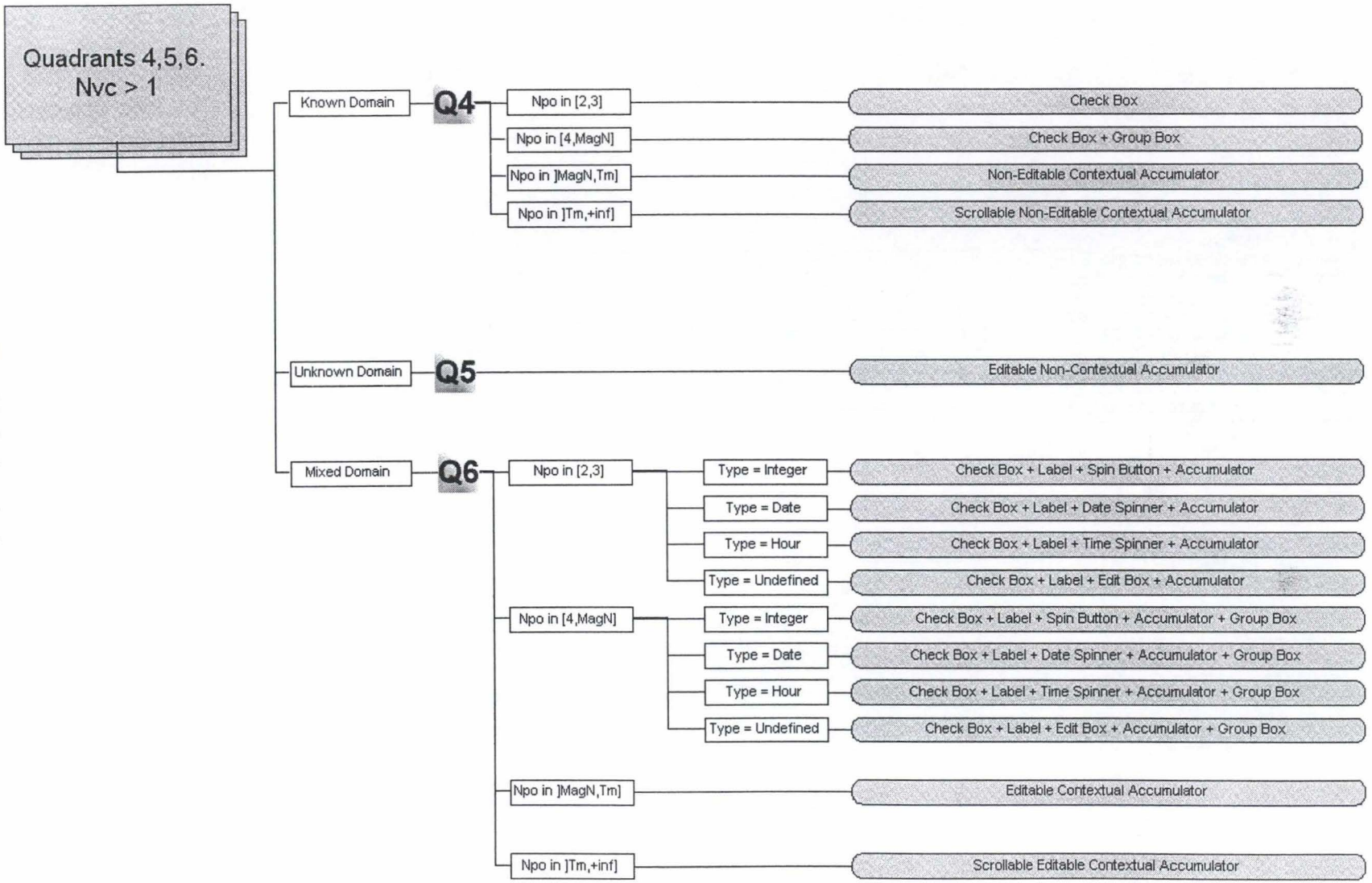


Figure B.2: Quadrants 4 to 6

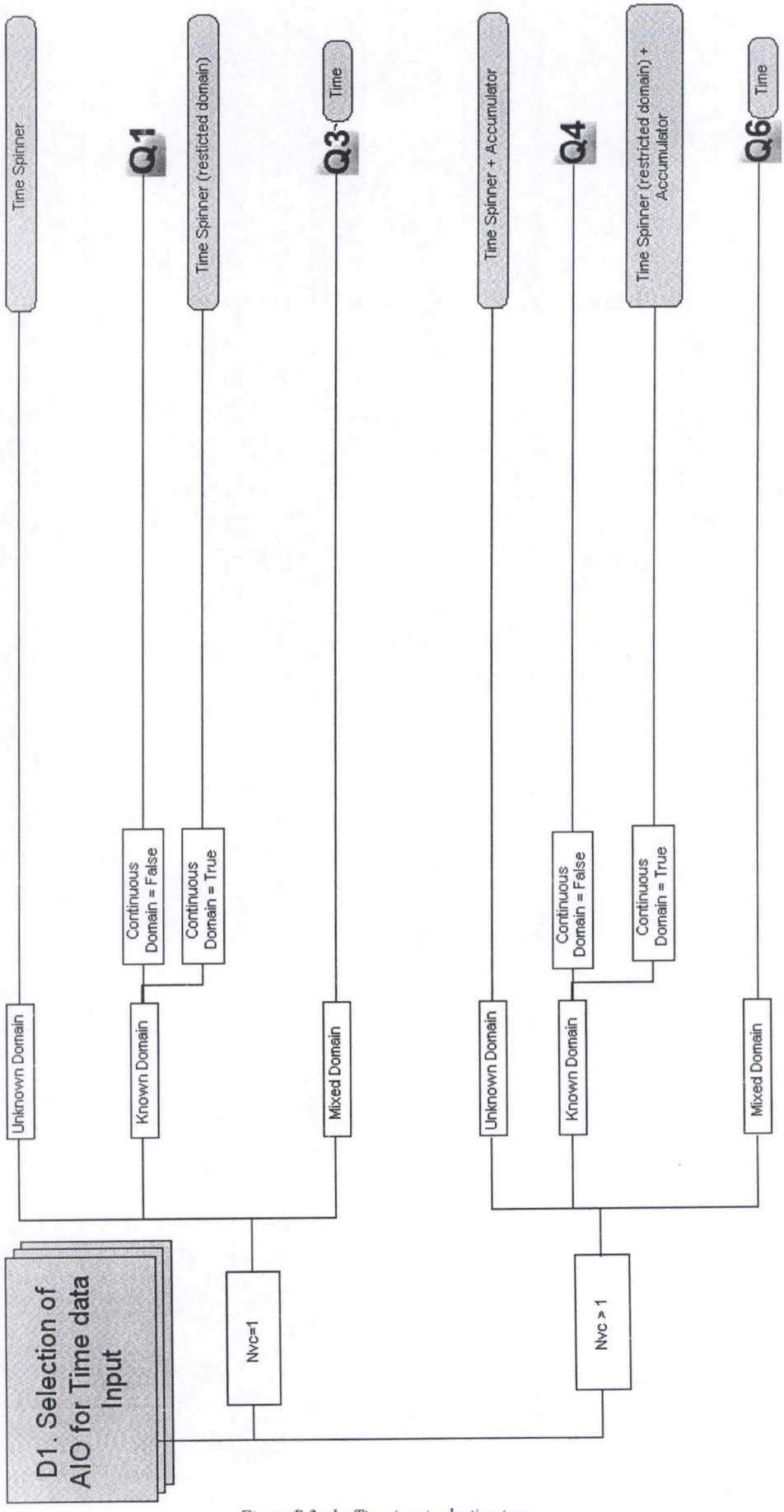


Figure B.3: the Time input selection tree

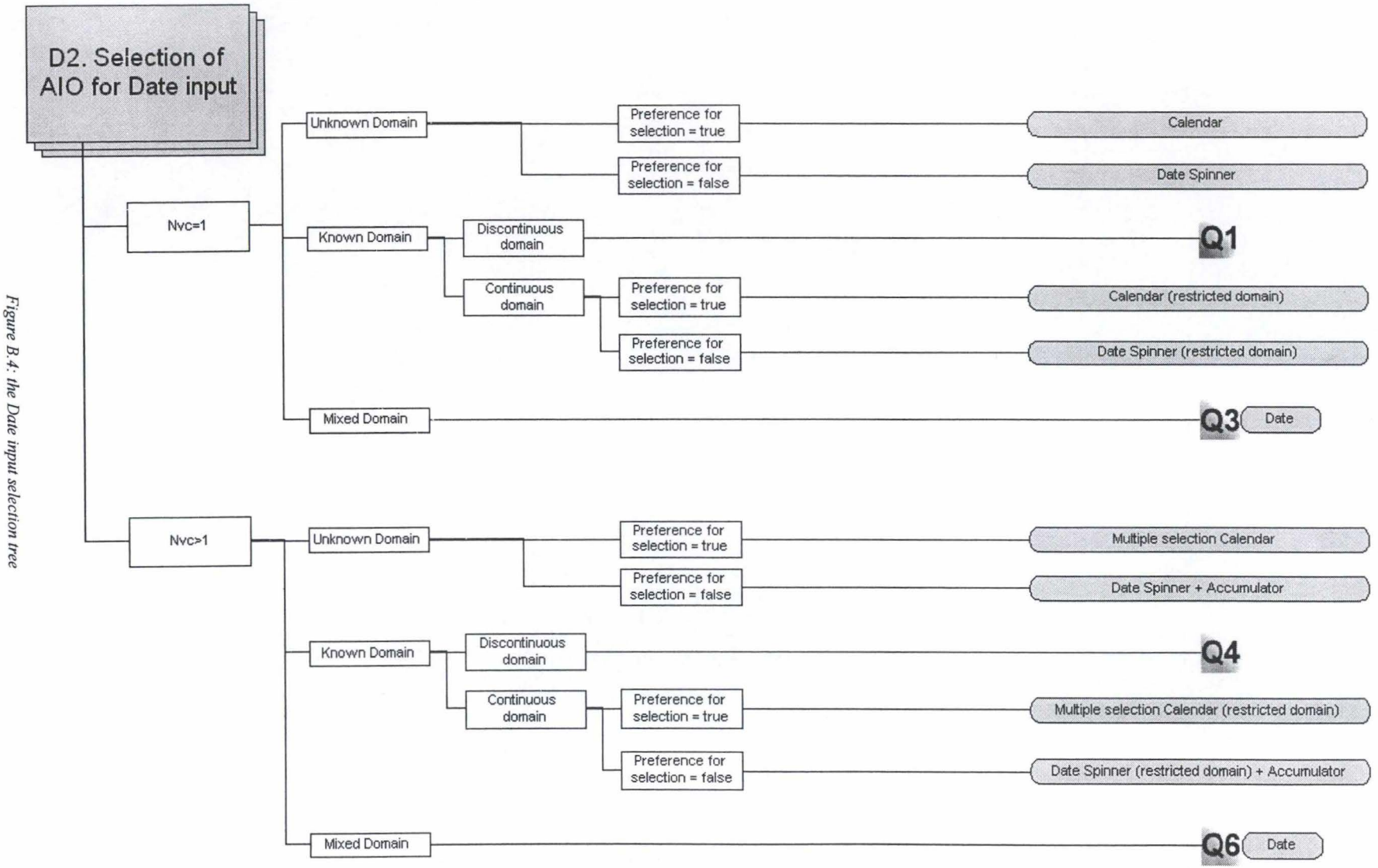


Figure B.4: the Date input selection tree

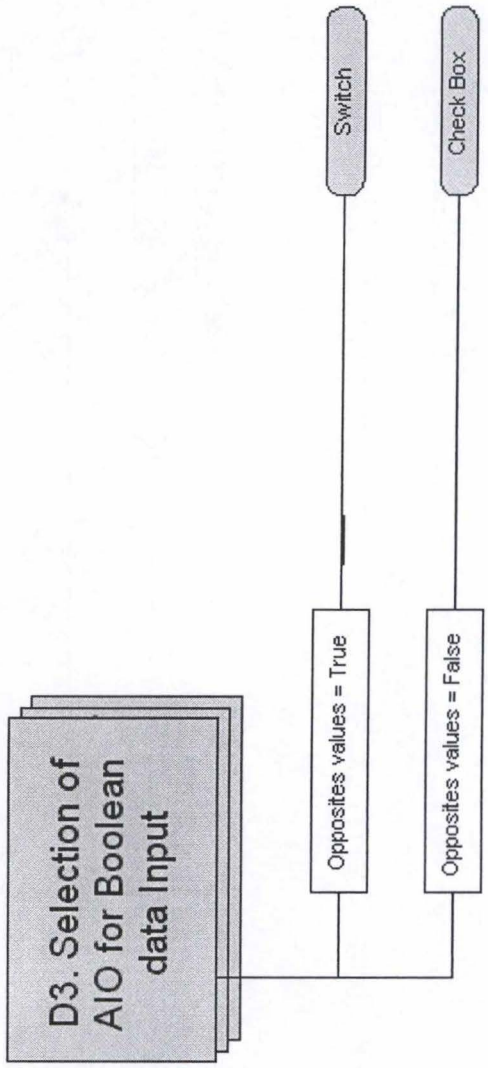


Figure B.5: the Boolean input selection tree

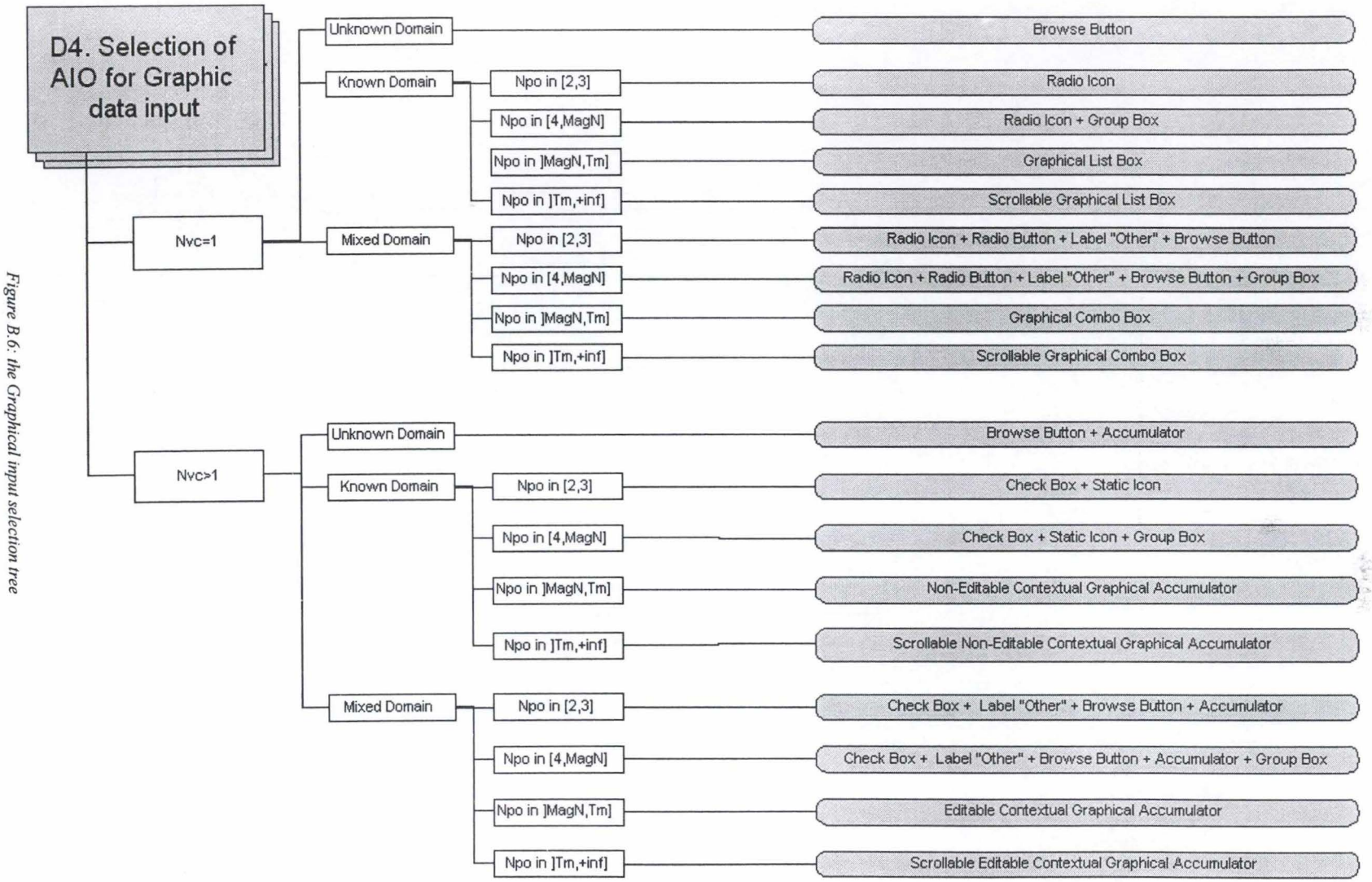


Figure B.6: the Graphical input selection tree

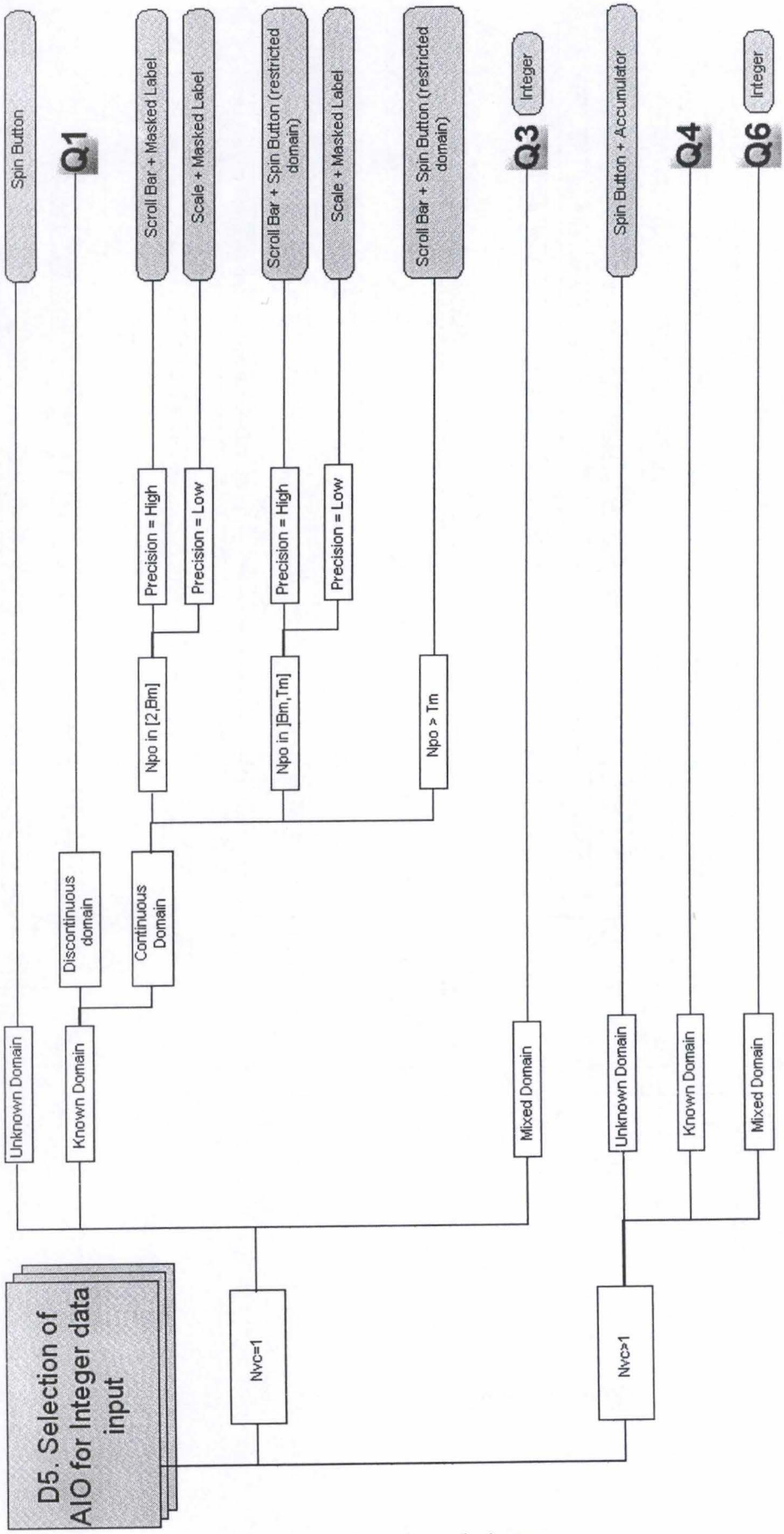


Figure B.7.: the *Integer* input selection tree

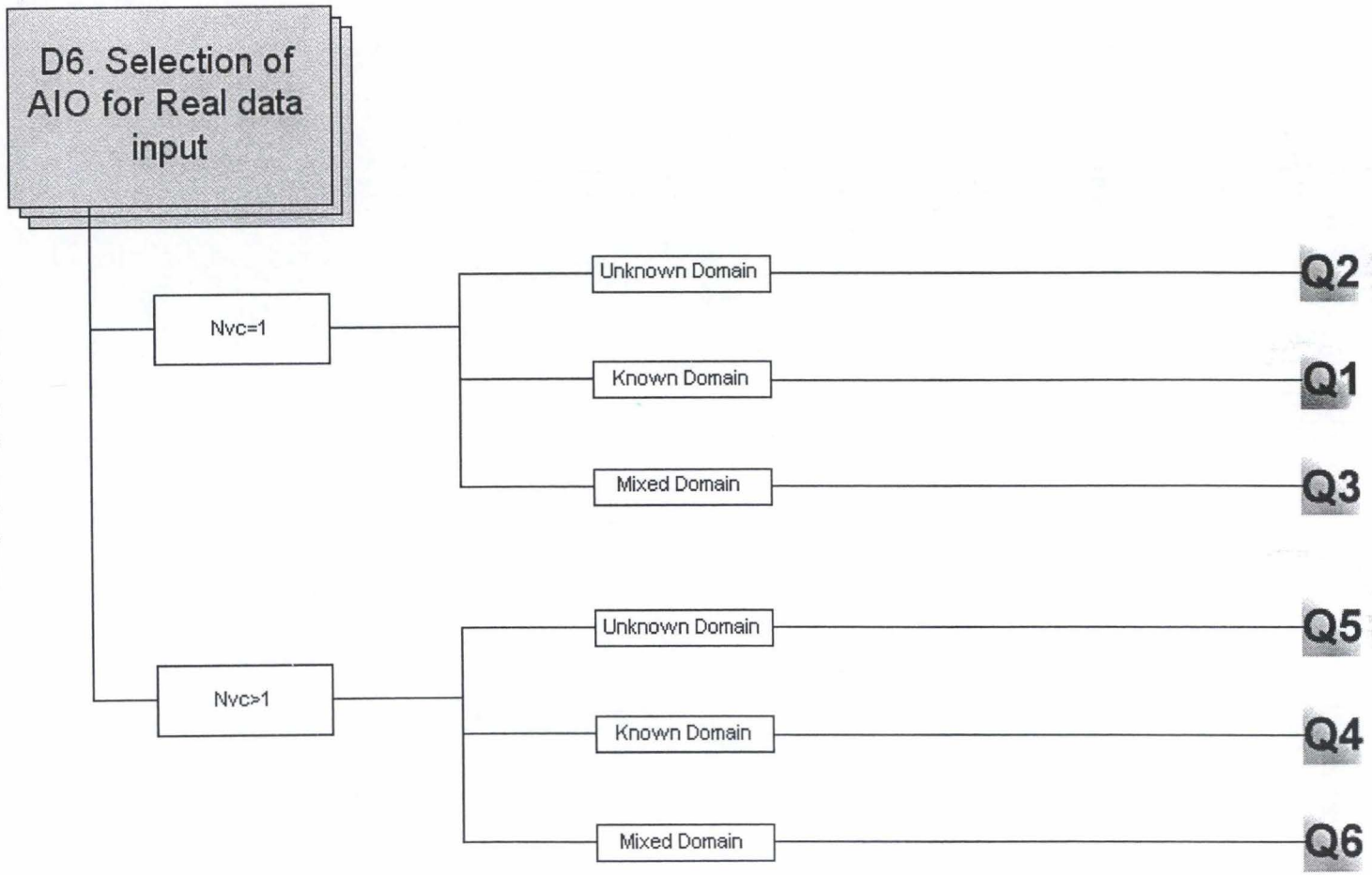


Figure B.8: the *Real* input selection tree

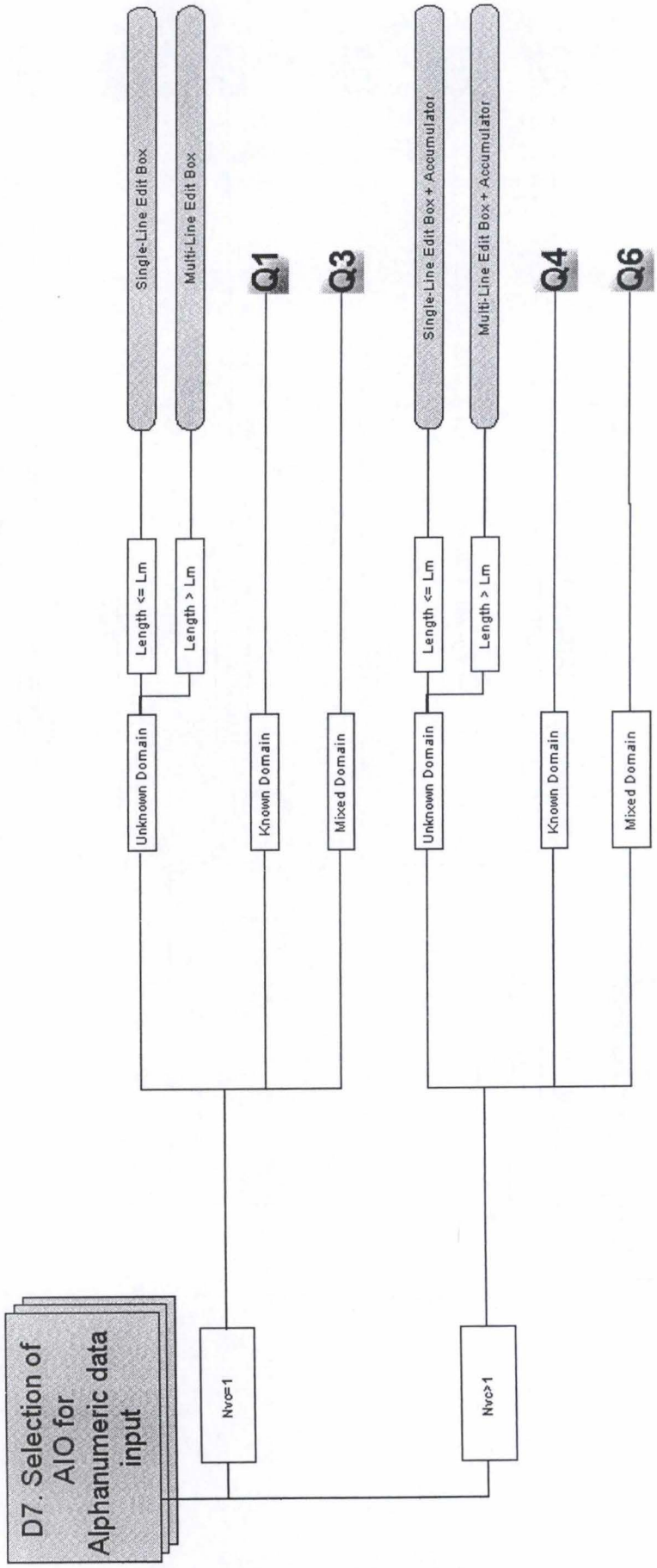


Figure B.9: the Alphanumeric input selection tree

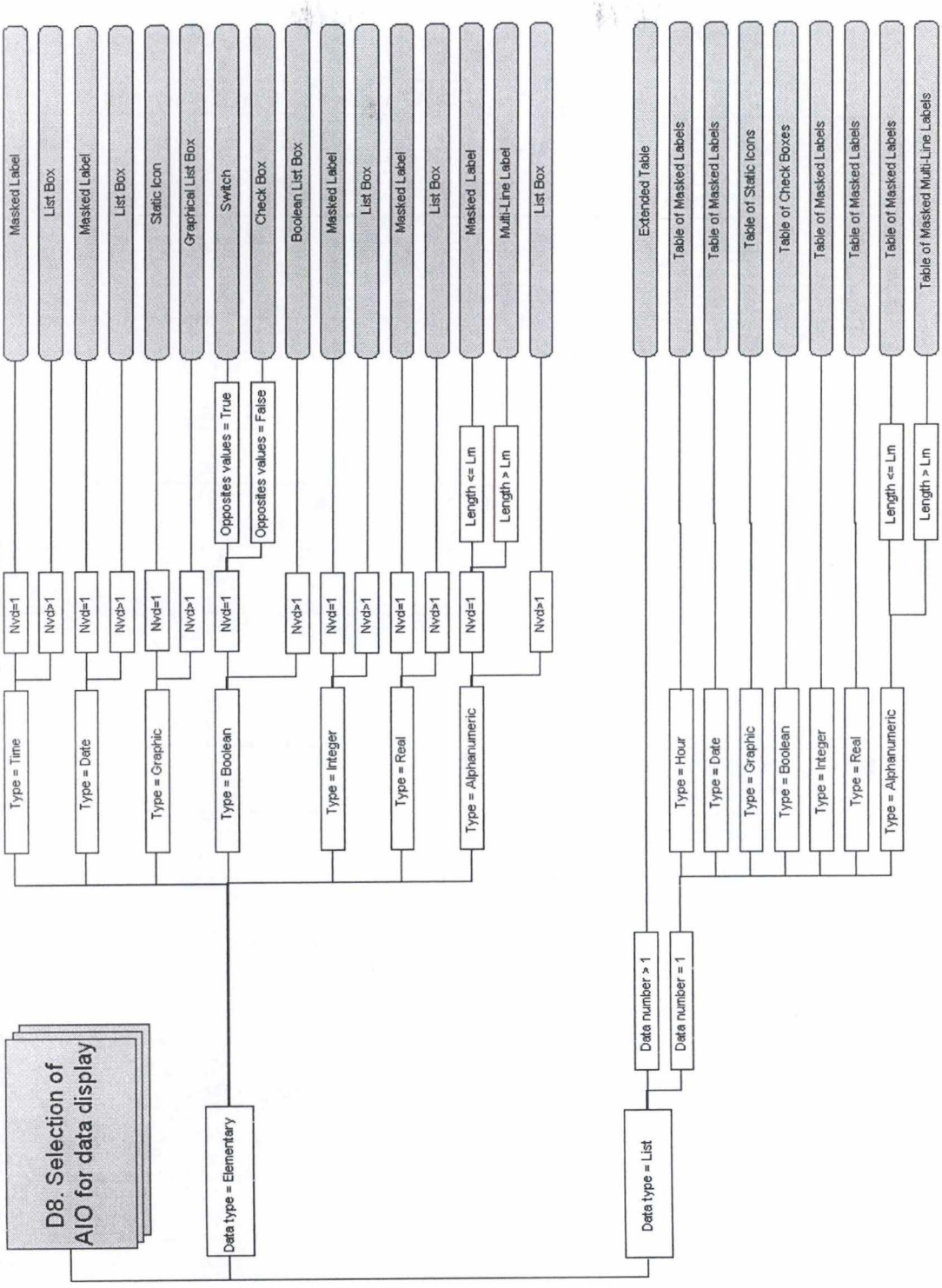


Figure B.10 : The Display Selection *tree*

8.3. AIOs and their alternatives

Num	AIOs	Lower Density alternatives
1	Boolean List Box	
2	Browse Button	
3	Browse Button + Accumulator	
4	Calendar	Drop-Down Calendar
5	Calendar (Restricted domain)	Drop-Down Calendar
6	Check Box	
7	Check Box + Group Box	
8	Check Box + Label "Other" + Browse Button + Accumulator	
9	Check Box + Label "Other" + Browse Button + Accumulator + Group Box	
10	Check Box + Label + Date Spinner + Accumulator	
11	Check Box + Label + Date Spinner + Accumulator + Group Box	
12	Check Box + Label + Edit Box + Accumulator	
13	Check Box + Label + Edit Box + Accumulator + Group Box	
14	Check Box + Label + Spin Button + Accumulator	
15	Check Box + Label + Spin Button + Accumulator + Group Box	
16	Check Box + Label + Time Spinner + Accumulator	
17	Check Box + Label + Time Spinner + Accumulator + Group Box	
18	Check Box + Static Icon	
19	Check Box + Static Icon + Group Box	
20	Combo Box	Drop-Down Combo Box
21	Date Spinner	
22	Date Spinner (Restricted domain)	
23	Date Spinner (Restricted domain) + Accumulator	
24	Date Spinner + Accumulator	
25	Drop-down Calendar	
26	Drop-Down Graphical Combo Box	
27	Drop-Down Graphical List Box	
28	Drop-Down List Box	
29	Drop-Down Scrollable Combo Box	
30	Drop-Down Scrollable Graphical Combo Box	
31	Drop-Down Scrollable Graphical List Box	
32	Drop-Down Scrollable List Box	

33	Editable Contextual Accumulator	Multiple Selection Combo Box
34	Editable Contextual Graphical Accumulator	
35	Extended Table	
36	Graphical Combo Box	Drop-Down Graphical Combo Box
37	Graphical List Box	Drop-Down Graphical List Box
38	List Box	Drop-Down List Box
39	Masked Label	
40	Masked Multi-Line Label	
41	Multi-Line Edit Box	
42	Multi-Line Edit Box + Accumulator	
43	Multiple Selection Calendar	
44	Multiple Selection Calendar (Restricted domain)	
45	Multiple Selection Combo Box	
46	Multiple Selection Graphical List Box	
47	Multiple Selection List Box	
48	Non-Editable Contextual Accumulator	Boolean List Box / Multiple Selection List Box
49	Non-Editable Contextual Graphical Accumulator	Multiple Selection Graphical List Box
50	Radio Button	Drop-Down List Box
51	Radio Button + Group Box	Drop-Down List Box
52	Radio Button + Label + Date Spinner	Drop-Down Combo Box
53	Radio Button + Label + Date Spinner + Group Box	Drop-Down Combo Box
54	Radio Button + Label + Edit Box	Drop-Down Combo Box
55	Radio Button + Label + Edit Box + Group Box	Drop-Down Combo Box
56	Radio Button + Label + Spin Button	Drop-Down Combo Box
57	Radio Button + Label + Spin Button + Group Box	Drop-Down Combo Box
58	Radio Button + Label + Time Spinner	Drop-Down Combo Box
59	Radio Button + Label + Time Spinner + Group Box	Drop-Down Combo Box
60	Radio Icon	Drop-Down Graphical List Box
61	Radio Icon + Group Box	Drop-Down Graphical List Box
62	Radio Icon + Radio Button + Label "Other" + Browse Button	Drop-Down Graphical Combo Box
63	Radio Icon + Radio Button + Label "Other" + Browse Button + Group Box	Drop-Down Graphical Combo Box
64	Scale + Masked Label	
65	Scroll Bar + Masked Label	
66	Scroll Bar + Spin Button (Restricted Domain)	
67	Scrollable Boolean List Box	
68	Scrollable Combo Box	Drop-Down Scrollable Combo Box
69	Scrollable Editable Contextual Accumula-	Scrollable Multiple Selection Combo

	Accumulator	Box
70	Scrollable Editable Contextual Graphical Accumulator	
71	Scrollable Graphical Combo Box	Drop-Down Scrollable Graphical Combo Box
72	Scrollable Graphical List Box	Drop-Down Scrollable Graphical List Box
73	Scrollable List Box	Drop-Down Scrollable List Box
74	Scrollable Multiple Selection Combo Box	
75	Scrollable Multiple Selection Graphical List Box	
76	Scrollable Multiple Selection List Box	
77	Scrollable Non-Editable Contextual Accumulator	Scrollable Boolean List Box / Scrollable Multiple Selection List Box
78	Scrollable Non-Editable Contextual Graphical Accumulator	Scrollable Multiple Selection Graphical List Box
79	Single-Line Edit Box	
80	Single-Line Edit Box + Accumulator	
81	Spin Button	
82	Spin Button + Accumulator	
83	Static Icon	
84	Switch	
85	Table of Check Boxes	
86	Table of Masked Labels	
87	Table of Masked Multi-Line Labels	
88	Table of Static Icons	
89	Time Spinner	
90	Time Spinner (restricted domain)	
91	Time Spinner (restricted domain) + Accumulator	
92	Time Spinner + Accumulator	

Table C.1: AIOs and their alternatives

8.4. High Level CIOs and their implemented AIOs

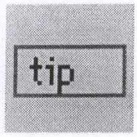
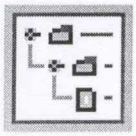
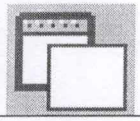
Num	High Level CIOs	*AIOs*
1	Boolean List Box	Boolean List Box Scrollable Boolean List Box
2	Browse Button	Browse Button
3	Browse Button Accumulator	Browse Button + Accumulator
4	Calendar	Calendar Calendar (Restricted domain) Multiple Selection Calendar Multiple Selection Calendar (Restricted domain)
5	Check Box Group	Check Box Check Box + Group Box Check Box + Static Icon Check Box + Static Icon + Group Box
6	Combo Box	Combo Box Multiple Selection Combo Box Scrollable Combo Box Scrollable Multiple Selection Combo Box
7	Combo Box Accumulator	Editable Contextual Accumulator Scrollable Editable Contextual Accumulator
8	Date Check Box Group	Check Box + Label + Date Spinner + Accumulator Check Box + Label + Date Spinner + Accumulator + Group Box
9	Date Radio Group	Radio Button + Label + Date Spinner Radio Button + Label + Date Spinner + Group Box
10	Date Spinner	Date Spinner Date Spinner (Restricted domain)
11	Date Spinner Accumulator	Date Spinner (Restricted domain) + Accumulator Date Spinner + Accumulator
12	Drop-Down Calendar	Drop-down Calendar
13	Drop-Down Combo Box	Drop-Down Scrollable Combo Box
14	Drop-Down Graphical Combo Box	Drop-Down Graphical Combo Box Drop-Down Scrollable Graphical Combo Box
15	Drop-Down List Box	Drop-Down Graphical List Box Drop-Down List Box Drop-Down Scrollable Graphical List Box Drop-Down Scrollable List Box
16	Edit Box	Single-Line Edit Box
17	Edit Box Accumulator	Single-Line Edit Box + Accumulator
18	Graphical Check Box Group	Check Box + Label "Other" + Browse Button + Accumulator Check Box + Label "Other" + Browse Button + Accumulator + Group Box

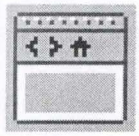
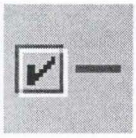
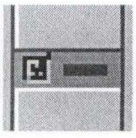
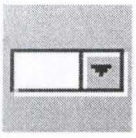
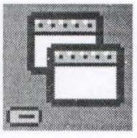
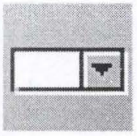
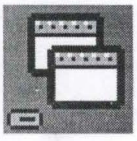
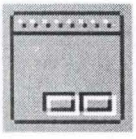
19	Graphical Combo Box	Graphical Combo Box Scrollable Graphical Combo Box
20	Graphical Combo Box Accumulator	Editable Contextual Graphical Accumulator Scrollable Editable Contextual Graphical Accumulator
21	Graphical Radio Group	Radio Icon + Radio Button + Label "Other" + Browse Button Radio Icon + Radio Button + Label "Other" + Browse Button + Group Box
22	List Box	Graphical List Box List Box Multiple Selection Graphical List Box Multiple Selection List Box Scrollable Graphical List Box Scrollable List Box Scrollable Multiple Selection Graphical List Box Scrollable Multiple Selection List Box
23	List Box Accumulator	Non-Editable Contextual Accumulator Non-Editable Contextual Graphical Accumulator Scrollable Non-Editable Contextual Accumulator Scrollable Non-Editable Contextual Graphical Accumulator
24	Masked Label	Masked Label
25	Multi-Line Edit Box	Multi-Line Edit Box
26	Multi-Line Edit Box Accumulator	Multi-Line Edit Box + Accumulator
27	MultiLine Label	Masked Multi-Line Label
28	Numeric Check Box Group	Check Box + Label + Spin Button + Accumulator Check Box + Label + Spin Button + Accumulator + Group Box
29	Numeric Radio Group	Radio Button + Label + Spin Button Radio Button + Label + Spin Button + Group Box
30	Numeric ScrollBar	Scroll Bar + Masked Label Scroll Bar + Spin Button (Restricted Domain)
31	Radio Group	Radio Button Radio Button + Group Box Radio Icon Radio Icon + Group Box
32	Scale	Scale + Masked Label
33	Spin Button	Spin Button
34	Spin Button Accumulator	Spin Button + Accumulator
35	Static Icon	Static Icon
36	Switch	Switch
37	Table	Extended Table Table of Check Boxes Table of Masked Labels Table of Masked Multi-Line Labels

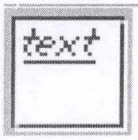
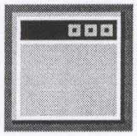
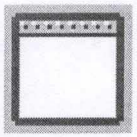
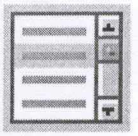
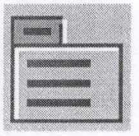
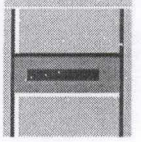
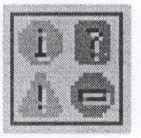
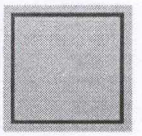
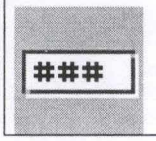
		Table of Static Icons
38	Text Check Box Group	Check Box + Label + Edit Box + Accumulator Check Box + Label + Edit Box + Accumulator + Group Box
39	Text Radio Group	Radio Button + Label + Edit Box Radio Button + Label + Edit Box + Group Box
40	Time Check Box Group	Check Box + Label + Time Spinner + Accumulator Check Box + Label + Time Spinner + Accumulator + Group Box
41	Time Radio Group	Radio Button + Label + Time Spinner Radio Button + Label + Time Spinner + Group Box
42	Time Spinner	Time Spinner Time Spinner (restricted domain)
43	Time Spinner Accumulator	Time Spinner (restricted domain) + Accumulator Time Spinner + Accumulator

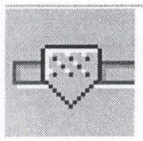
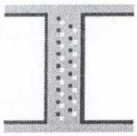
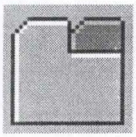
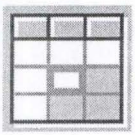
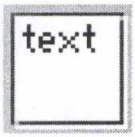
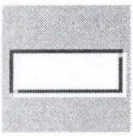
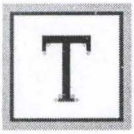
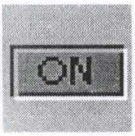
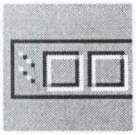
Table D.1: High Level CIOs and their implemented AIOs

8.5. Overview of the Swing components

JFC Component	Code Name	Common Name
	`JToolTip`	Tool tip
	`JTree`	Tree view
	`JWindow`	Plain (undadorned) window

JFC Component	Code Name	Common Name
	`JApplet`	Applet
	`JButton`	Command button
	`JCheckBox`	Checkbox
	`JCheckBoxMenuItem`	Checkbox menu item
	`JColorChooser`	Color chooser
	`JComboBox`	Noneditable and editable combo boxes
	`JDesktopPane`	Desktop pane
	`JComboBox`	Noneditable and editable combo boxes
	`JDesktopPane`	Desktop pane
	`JDialog`	Dialog box, secondary window, and utility window

JFC Component	Code Name	Common Name
	JEditorPane	Editor pane
	JFrame	Primary window
	JInternalFrame	Internal frame, minimized internal frame, and palette
	JLabel	Label
	JList	List
	JMenu	Drop-down menu and submenu
	JMenuBar	Menu bar
	JMenuItem	Menu item
	JOptionPane	Alert box
	JPanel	Panel
	JPasswordField	Password field

JFC Component	Code Name	Common Name
	`JSlider`	Slider
	`JSplitPane`	Split pane
	`JTabbedPane`	Tabbed pane
	`JTable`	Table
	`JTextArea`	Plain text ar
	`JTextField`	Editable and noneditable field (single
	`JTextPane`	Formatted te pane
	`JToggleButton`	Toggle butto
	`JToolBar`	Toolbar

8.6. Selection rules spreadsheets

Num	AIO	Data Type	Type	Data Number	Nvd	Lg	Opposites Values
1	Masked Label	Elementary	Hour		Simple (=1)		
2	List Box	Elementary	Hour		Multiple (>1)		
3	Masked Label	Elementary	Date		Simple (=1)		
4	List Box	Elementary	Date		Multiple (>1)		
5	Static Icon	Elementary	Graphic		Simple (=1)		
6	Graphic List Box	Elementary	Graphic		Multiple (>1)		
7	Switch	Elementary	Boolean		Simple (=1)		VRAI
8	Check Box	Elementary	Boolean		Simple (=1)		FAUX
9	Boolean List Box	Elementary	Boolean		Multiple (>1)		
10	Masked Label	Elementary	Integer		Simple (=1)		
11	List Box	Elementary	Integer		Multiple (>1)		
12	Masked Label	Elementary	Real		Simple (=1)		
13	List Box	Elementary	Real		Multiple (>1)		
14	Masked Label	Elementary	Alphanumeric		Simple (=1)	<=Lm	
15	Masked Multi-Line Label	Elementary	Alphanumeric		Simple (=1)	> Lm	
16	List Box	Elementary	Alphanumeric		Multiple (>1)		
17	Extended Table	List		>1			
18	Table of Masked Labels	List	Hour	1			
19	Table of Masked Labels	List	Date	1			
20	Table of Static Icons	List	Graphic	1			
21	Table of Check Boxes	List	Boolean	1			
22	Table of Masked Labels	List	Integer	1			
23	Table of Masked Labels	List	Real	1			
24	Table of Masked Labels	List	Alphanumeric	1		<= Lm	
25	Table of Masked Multi-Line Labels	List	Alphanumeric	1		> Lm	

Table 8.1 Display Selection Rules

Appendix F : INPUT Selection Rules

Num	AIO	Type	Domain	Choice	Npo	Cont	Lg	Precision	Preference for selection	Opposites values
1	Time Spinner	Time	Unknown	Simple(Nvc = 1)						
2	Radio Button	Time	Known	Simple(Nvc = 1)	[2,3]	FALSE				
3	Radio Button + Group Box	Time	Known	Simple(Nvc = 1)	[4,MagN]	FALSE				
4	List Box	Time	Known	Simple(Nvc = 1)	]MagN,Tm]	FALSE				
5	Scrollable List Box	Time	Known	Simple(Nvc = 1)	]Tm,+inf]	FALSE				
6	Time Spinner (restricted domain)	Time	Known	Simple(Nvc = 1)		TRUE				
7	Radio Button + Label + Time Spinner	Time	Mixed	Simple(Nvc = 1)	[2,3]					
8	Radio Button + Label + Time Spinner + Group Box	Time	Mixed	Simple(Nvc = 1)	[4,MagN]					
9	Combo Box	Time	Mixed	Simple(Nvc = 1)	]MagN,Tm]					
10	Scrollable Combo Box	Time	Mixed	Simple(Nvc = 1)	]Tm,+inf]					
11	Time Spinner + Accumulator	Time	Unknown	Multiple(Nvc > 1)						
12	Check Box	Time	Known	Multiple(Nvc > 1)	[2,3]	FALSE				
13	Check Box + Group Box	Time	Known	Multiple(Nvc > 1)	[4,MagN]	FALSE				
14	Non-Editable Contextual Accumulator	Time	Known	Multiple(Nvc > 1)	]MagN,Tm]	FALSE				
15	Scrollable Non-Editable Contextual Accumulator	Time	Known	Multiple(Nvc > 1)	]Tm,+inf]	FALSE				
16	Time Spinner (restricted domain) + Accumulator	Time	Known	Multiple(Nvc > 1)		TRUE				
17	Check Box + Label + Time Spinner + Accumulator	Time	Mixed	Multiple(Nvc > 1)	[2,3]					
18	Check Box + Label + Time Spinner + Accumulator + Group Box	Time	Mixed	Multiple(Nvc > 1)	[4,MagN]					
19	Editable Contextual Accumulator	Time	Mixed	Multiple(Nvc > 1)	]MagN,Tm]					
20	Scrollable Editable Contextual Accumulator	Time	Mixed	Multiple(Nvc > 1)	]Tm,+inf]					
21	Calendar	Date	Unknown	Simple(Nvc = 1)					TRUE	
22	Date Spinner	Date	Unknown	Simple(Nvc = 1)					FALSE	
23	Radio Button	Date	Known	Simple(Nvc = 1)	[2,3]	FALSE				
24	Radio Button + Group Box	Date	Known	Simple(Nvc = 1)	[4,MagN]	FALSE				
25	List Box	Date	Known	Simple(Nvc = 1)	]MagN,Tm]	FALSE				
26	Scrollable List Box	Date	Known	Simple(Nvc = 1)	]Tm,+inf]	FALSE				
27	Calendar (Restricted domain)	Date	Known	Simple(Nvc = 1)		TRUE			TRUE	
28	Date Spinner (Restricted domain)	Date	Known	Simple(Nvc = 1)		TRUE			FALSE	
29	Radio Button + Label + Date Spinner	Date	Mixed	Simple(Nvc = 1)	[2,3]					
30	Radio Button + Label + Date Spinner + Group Box	Date	Mixed	Simple(Nvc = 1)	[4,MagN]					
31	Combo Box	Date	Mixed	Simple(Nvc = 1)	]MagN,Tm]					
32	Scrollable Combo Box	Date	Mixed	Simple(Nvc = 1)	]Tm,+inf]					
33	Multiple Selection Calendar	Date	Unknown	Multiple(Nvc > 1)					TRUE	
34	Date Spinner + Accumulator	Date	Unknown	Multiple(Nvc > 1)					FALSE	
35	Check Box	Date	Known	Multiple(Nvc > 1)	[2,3]	FALSE				
36	Check Box + Group Box	Date	Known	Multiple(Nvc > 1)	[4,MagN]	FALSE				
37	Non-Editable Contextual Accumulator	Date	Known	Multiple(Nvc > 1)	]MagN,Tm]	FALSE				
38	Scrollable Non-Editable Contextual Accumulator	Date	Known	Multiple(Nvc > 1)	]Tm,+inf]	FALSE				
39	Multiple Selection Calendar (Restricted domain)	Date	Known	Multiple(Nvc > 1)		TRUE			TRUE	
40	Date Spinner (Restricted domain) + Accumulator	Date	Known	Multiple(Nvc > 1)		TRUE			FALSE	
41	Check Box + Label + Date Spinner + Accumulator	Date	Mixed	Multiple(Nvc > 1)	[2,3]					
42	Check Box + Label + Date Spinner + Accumulator + Group Box	Date	Mixed	Multiple(Nvc > 1)	[4,MagN]					
43	Editable Contextual Accumulator	Date	Mixed	Multiple(Nvc > 1)	]MagN,Tm]					
44	Scrollable Editable Contextual Accumulator	Date	Mixed	Multiple(Nvc > 1)	]Tm,+inf]					
45	Switch	Boolean								TRUE
46	Check Box	Boolean								FALSE
47	Browse Button	Graphical	Unknown	Simple(Nvc = 1)						
48	Radio Icon	Graphical	Known	Simple(Nvc = 1)	[2,3]					
49	Radio Icon + Group Box	Graphical	Known	Simple(Nvc = 1)	[4,MagN]					
50	Graphical List Box	Graphical	Known	Simple(Nvc = 1)	]MagN,Tm]					
51	Scrollable Graphical List Box	Graphical	Known	Simple(Nvc = 1)	]Tm,+inf]					
52	Radio Icon + Radio Button + Label "Other" + Browse Button	Graphical	Mixed	Simple(Nvc = 1)	[2,3]					
53	Radio Icon + Radio Button + Label "Other" + Browse Button + Grou	Graphical	Mixed	Simple(Nvc = 1)	[4,MagN]					
54	Graphical Combo Box	Graphical	Mixed	Simple(Nvc = 1)	]MagN,Tm]					
55	Scrollable Graphical Combo Box	Graphical	Mixed	Simple(Nvc = 1)	]Tm,+inf]					
56	Browse Button + Accumulator	Graphical	Unknown	Multiple(Nvc > 1)						

57	Check Box + Static Icon	Graphical	Known	Multiple(Nvc > 1)	[2,3]					
58	Check Box + Static Icon + Group Box	Graphical	Known	Multiple(Nvc > 1)	[4,MagN]					
59	Non-Editable Contextual Graphical Accumulator	Graphical	Known	Multiple(Nvc > 1)	]MagN,Tm]					
60	Scrollable Non-Editable Contextual Graphical Accumulator	Graphical	Known	Multiple(Nvc > 1)	]Tm,+inf]					
61	Check Box + Label "Other" + Browse Button + Accumulator	Graphical	Mixed	Multiple(Nvc > 1)	[2,3]					
62	Check Box + Label "Other" + Browse Button + Accumulator + Grou	Graphical	Mixed	Multiple(Nvc > 1)	[4,MagN]					
63	Editable Contextual Graphical Accumulator	Graphical	Mixed	Multiple(Nvc > 1)	]MagN,Tm]					
64	Scrollable Editable Contextual Graphical Accumulator	Graphical	Mixed	Multiple(Nvc > 1)	]Tm,+inf]					
65	Single-Line Edit Box	Alphanumer	Unknown	Simple (Nvc=1)			<=Lm			
66	Multi-Line Edit Box	Alphanumer	Unknown	Simple (Nvc=1)			>Lm			
67	Radio Button	Alphanumer	Known	Simple (Nvc=1)	[2,3]					
68	Radio Button + Group Box	Alphanumer	Known	Simple (Nvc=1)	[4,MagN]					
69	List Box	Alphanumer	Known	Simple (Nvc=1)	]MagN,Tm]					
70	Scrollable List Box	Alphanumer	Known	Simple (Nvc=1)	]Tm,+inf]					
71	Radio Button + Label + Edit Box	Alphanumer	Mixed	Simple (Nvc=1)	[2,3]					
72	Radio Button + Label + Edit Box + Group Box	Alphanumer	Mixed	Simple (Nvc=1)	[4,MagN]					
73	Combo Box	Alphanumer	Mixed	Simple (Nvc=1)	]MagN,Tm]					
74	Scrollable Combo Box	Alphanumer	Mixed	Simple (Nvc=1)	]Tm,+inf]					
75	Single-Line Edit Box + Accumulator	Alphanumer	Unknown	Multiple (Nvc>1)			<=Lm			
76	Multi-Line Edit Box + Accumulator	Alphanumer	Unknown	Multiple (Nvc>1)			>Lm			
77	Check Box	Alphanumer	Known	Multiple (Nvc>1)	[2,3]					
78	Check Box + Group Box	Alphanumer	Known	Multiple (Nvc>1)	[4,MagN]					
79	Non-Editable Contextual Accumulator	Alphanumer	Known	Multiple (Nvc>1)	]MagN,Tm]					
80	Scrollable Non-Editable Contextual Accumulator	Alphanumer	Known	Multiple (Nvc>1)	]Tm,+inf]					
81	Check Box + Label + Edit Box + Accumulator	Alphanumer	Mixed	Multiple (Nvc>1)	[2,3]					
82	Check Box + Label + Edit Box + Accumulator + Group Box	Alphanumer	Mixed	Multiple (Nvc>1)	[4,MagN]					
83	Editable Contextual Accumulator	Alphanumer	Mixed	Multiple (Nvc>1)	]MagN,Tm]					
84	Scrollable Editable Contextual Accumulator	Alphanumer	Mixed	Multiple (Nvc>1)	]Tm,+inf]					
85	Edit Box	Real	Unknown	Simple (Nvc=1)						
86	Radio Button	Real	Known	Simple (Nvc=1)	[2,3]					
87	Radio Button + Group Box	Real	Known	Simple (Nvc=1)	[4,MagN]					
88	List Box	Real	Known	Simple (Nvc=1)	]MagN,Tm]					
89	Scrollable List Box	Real	Known	Simple (Nvc=1)	]Tm,+inf]					
90	Radio Button + Label + Edit Box	Real	Mixed	Simple (Nvc=1)	[2,3]					
91	Radio Button + Label + Edit Box + Group Box	Real	Mixed	Simple (Nvc=1)	[4,MagN]					
92	Combo Box	Real	Mixed	Simple (Nvc=1)	]MagN,Tm]					
93	Scrollable Combo Box	Real	Mixed	Simple (Nvc=1)	]Tm,+inf]					
94	Edit Box + Accumulator	Real	Unknown	Multiple (Nvc>1)						
95	Check Box	Real	Known	Multiple (Nvc>1)	[2,3]					
96	Check Box + Group Box	Real	Known	Multiple (Nvc>1)	[4,MagN]					
97	Non-Editable Contextual Accumulator	Real	Known	Multiple (Nvc>1)	]MagN,Tm]					
98	Scrollable Non-Editable Contextual Accumulator	Real	Known	Multiple (Nvc>1)	]Tm,+inf]					
99	Check Box + Label + Edit Box + Accumulator	Real	Mixed	Multiple (Nvc>1)	[2,3]					
100	Check Box + Label + Edit Box + Accumulator + Group Box	Real	Mixed	Multiple (Nvc>1)	[4,MagN]					
101	Editable Contextual Accumulator	Real	Mixed	Multiple (Nvc>1)	]MagN,Tm]					
102	Scrollable Editable Contextual Accumulator	Real	Mixed	Multiple (Nvc>1)	]Tm,+inf]					
103	Spin Button	Integer	Unknown	Simple (Nvc=1)						
104	Radio Button	Integer	Known	Simple (Nvc=1)	[2,3]	FALSE				
105	Radio Button + Group Box	Integer	Known	Simple (Nvc=1)	[4,MagN]	FALSE				
106	List Box	Integer	Known	Simple (Nvc=1)	]MagN,Tm]	FALSE				
107	Scrollable List Box	Integer	Known	Simple (Nvc=1)	]Tm,+inf]	FALSE				
108	Scroll Bar + Masked Label	Integer	Known	Simple (Nvc=1)	[2,Bm]	TRUE		High		
109	Scale + Masked Label	Integer	Known	Simple (Nvc=1)	[2,Bm]	TRUE		Low		
110	Scroll Bar + Spin Button (Restricted Domain)	Integer	Known	Simple (Nvc=1)	]Bm,Tm]	TRUE		High		
111	Scale + Masked Label	Integer	Known	Simple (Nvc=1)	]Bm,Tm]	TRUE		Low		
112	Scroll Bar + Spin Button (Restricted Domain)	Integer	Known	Simple (Nvc=1)	> Tm	TRUE				
113	Radio Button + Label + Spin Button	Integer	Mixed	Simple (Nvc=1)	[2,3]					
114	Radio Button + Label + Spin Button + Group Box	Integer	Mixed	Simple (Nvc=1)	[4,MagN]					
115	Combo Box	Integer	Mixed	Simple (Nvc=1)	]MagN,Tm]					
116	Scrollable Combo Box	Integer	Mixed	Simple (Nvc=1)	]Tm,+inf]					
117	Spin Button + Accumulator	Integer	Unknown	Multiple (Nvc>1)						
118	Check Box	Integer	Known	Multiple (Nvc>1)	[2,3]					
119	Check Box + Group Box	Integer	Known	Multiple (Nvc>1)	[4,MagN]					
120	Non-Editable Contextual Accumulator	Integer	Known	Multiple (Nvc>1)	]MagN,Tm]					
121	Scrollable Non-Editable Contextual Accumulator	Integer	Known	Multiple (Nvc>1)	]Tm,+inf]					
122	Check Box + Label + Spin Button + Accumulator	Integer	Mixed	Multiple (Nvc>1)	[2,3]					
123	Check Box + Label + Spin Button + Accumulator + Group Box	Integer	Mixed	Multiple (Nvc>1)	[4,MagN]					
124	Editable Contextual Accumulator	Integer	Mixed	Multiple (Nvc>1)	]MagN,Tm]					
125	Scrollable Editable Contextual Accumulator	Integer	Mixed	Multiple (Nvc>1)	]Tm,+inf]					

B

IOs Database

This section contains information concerning the IOs datatbase.

1. Conceptual schema

The figure B-1 shows the conceptual schema of the IOs database.

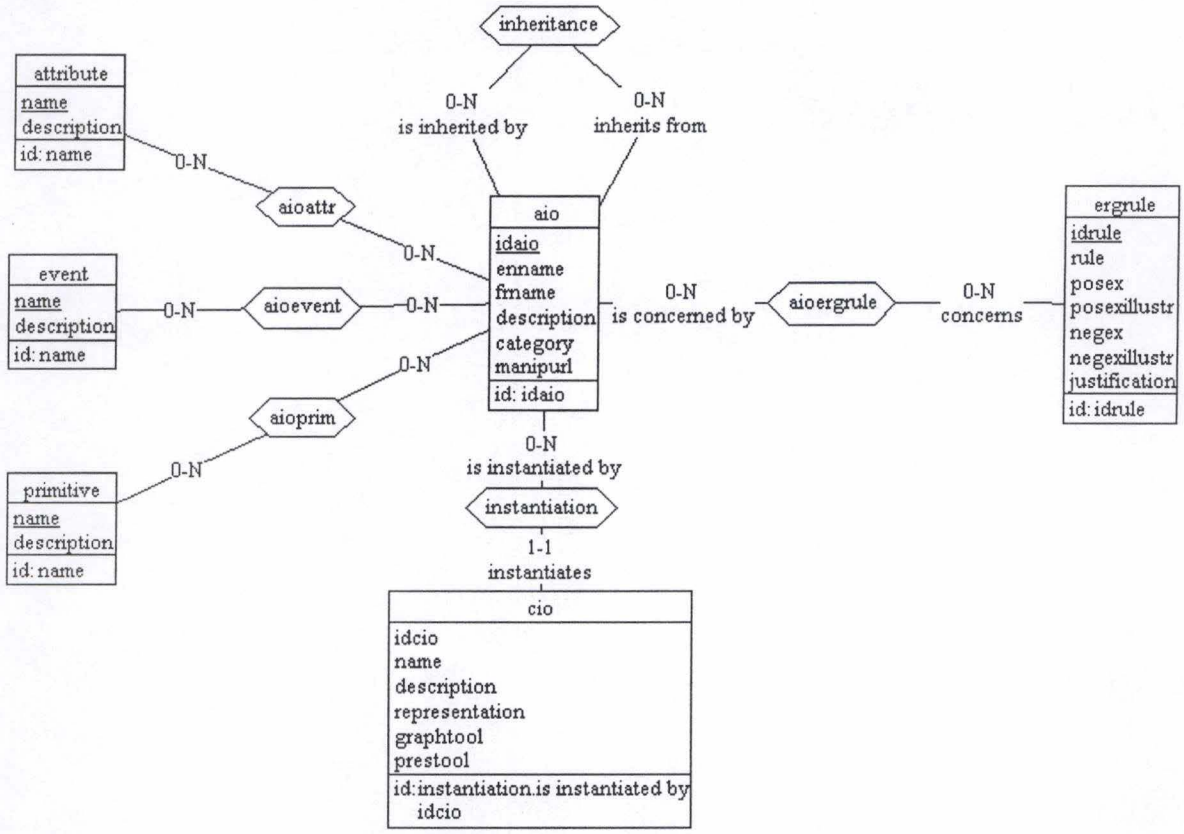


***Figure B-1**. The conceptual schema of the Ios database*

2. SQL database creation script

```
-- Database Section
-- ________________

create database IOs database;


-- DBSpace Section
-- _______________


-- Table Section
-- _____________

create table aio (
```

```
      idaio char(3) not null ,
      enname varchar2(50),
      frname varchar2(50) not null ,
      description long,
      category char(1) not null ,
      manipurl varchar2(50),
      primary key (idaio));

create table attribute (
      name varchar2(50) not null ,
      description long not null ,
      primary key (name));

create table cio (
      idaio char(3) not null ,
      idcio numeric(2) not null ,
      name varchar2(50) not null,
      description long,
      representation varchar2(50),
      graphtool varchar2(40) not null ,
      prestool varchar2(40),
      primary key (idaio, idcio) ,
      foreign key (idaio) references aio);

create table event (
      name varchar2(50) not null ,
      description long not null ,
      primary key (name));

create table inheritance (
      Inh_idaio char(3) not null ,
      idaio char(3) not null ,
      primary key (Inh_idaio, idaio) ,
      foreign key (idaio) references aio ,
      foreign key (Inh_idaio) references aio);

create table primitive (
      name varchar2(50) not null ,
      description long not null ,
      primary key (name));

create table aioattr (
      idaio char(3) not null ,
      name varchar2(50) not null ,
      primary key (idaio, name) ,
      foreign key (name) references attribute ,
      foreign key (idaio) references aio);

create table aioevent (
      idaio char(3) not null ,
      name varchar2(50) not null ,
```

```
    primary key (idaio, name) ,
    foreign key (name) references event ,
    foreign key (idaio) references aio);

create table aioprim (
    idaio char(3) not null ,
    name varchar2(50) not null ,
    primary key (idaio, name) ,
    foreign key (name) references primitive ,
    foreign key (idaio) references aio);

create table ergrule (
    idrule numeric(5) not null ,
    name varchar2(50) not null ,
    rule long not null,
    posex varchar2(300),
    posexillustr varchar2(50),
    negex varchar2(300),
    negexillustr varchar2(50),
    justification varchar2(300),
    primary key (idrule));

create table aiorule (
    idaio char(3) not null ,
    idrule numeric(5) not null ,
    primary key (idrule, idaio) ,
    foreign key (idrule) references ergrule ,
    foreign key (idaio) references aio);
```

C

Information Architecture

After having divided the course into information chunks, we have organized them into hierarchies. We have organized information according to two distinct scenarios.

In the first scenario (Objectivist), we have approximately followed the paper course structure. The two interactive applications take place in two distinct sections (Interactive Objects and AIOs selection).

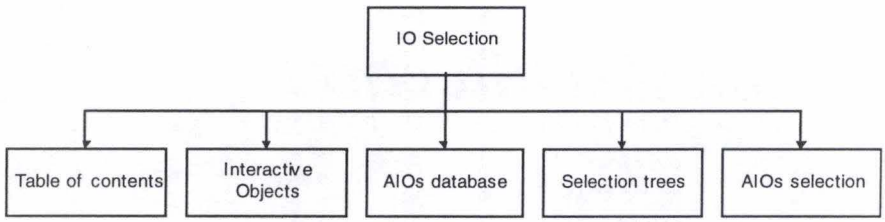
The second scenario (Constructivist) organizes the course into two main parts: the *interactive objects* and the *AIOs selection.* The first part begins with the IO manipulation application while the second part begins with the selection application.

Here, you will find the complete information architecture for the two scenarios. Each of them include the hierarchical and sequential navigation

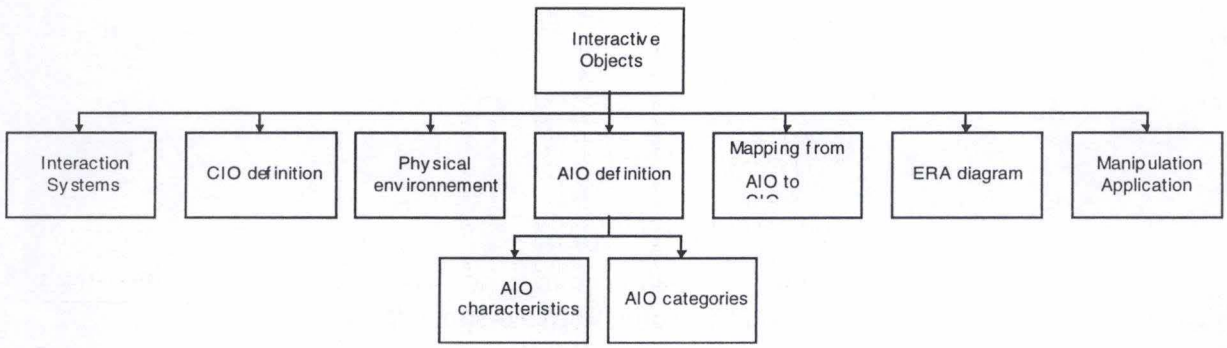
1. Objectivist Scenario

1.1. Hierarchical Navigation

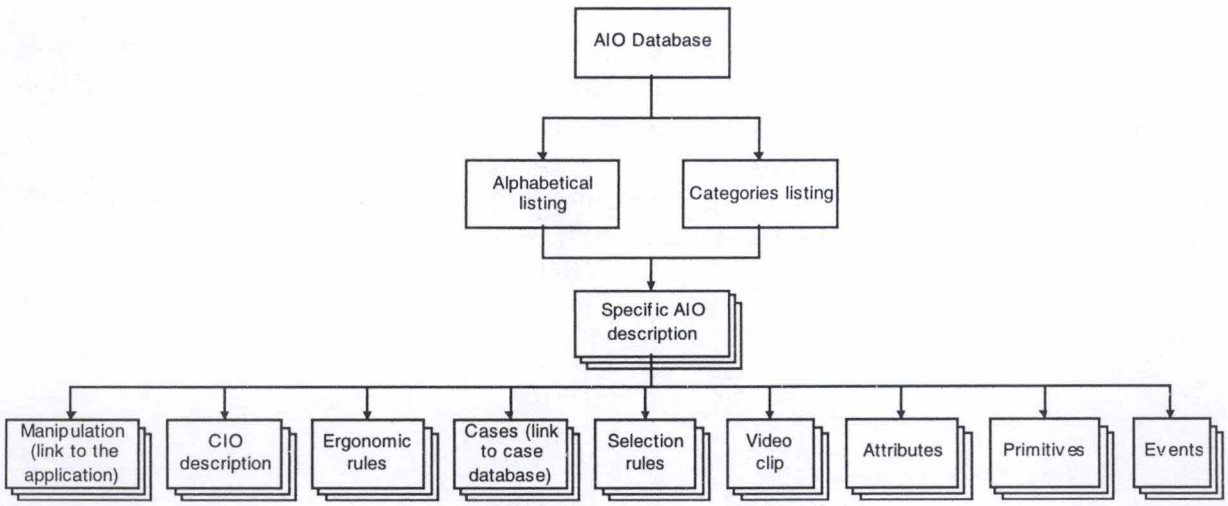
1.1.1. Top Level



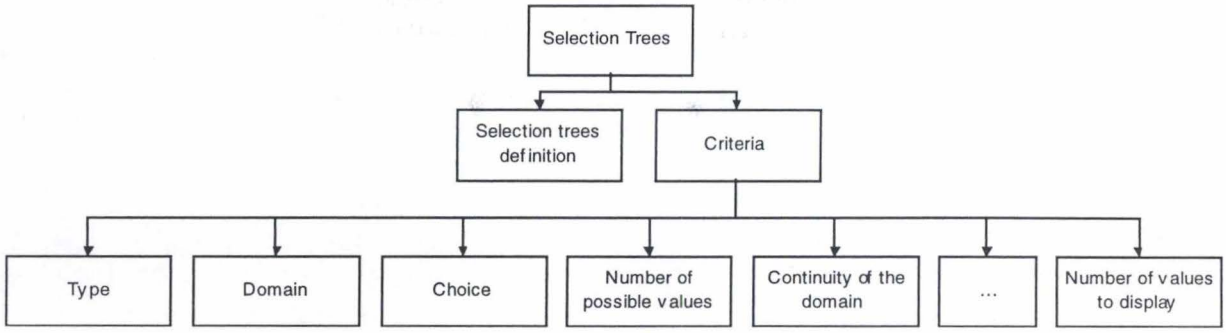
1.1.2. Interactive Objects Section



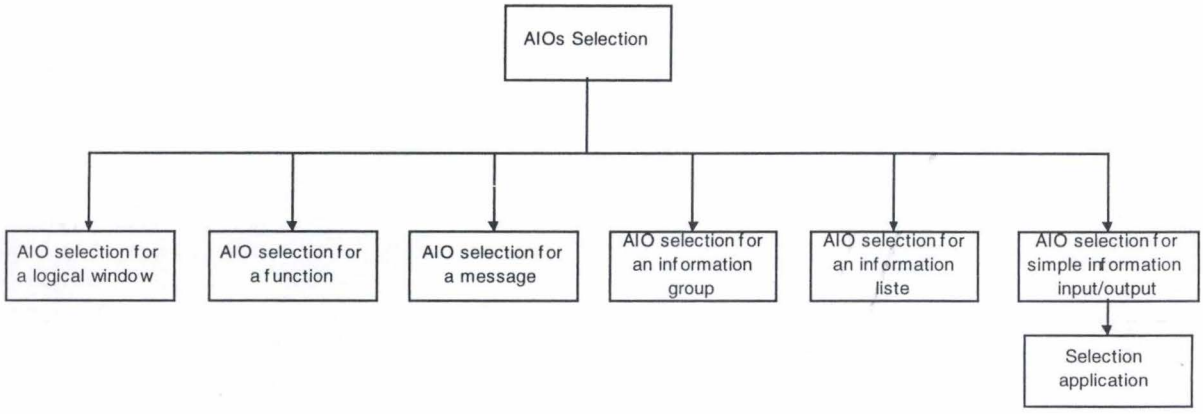
1.1.3. AIOs Database Section



1.1.4. Selection Trees Section

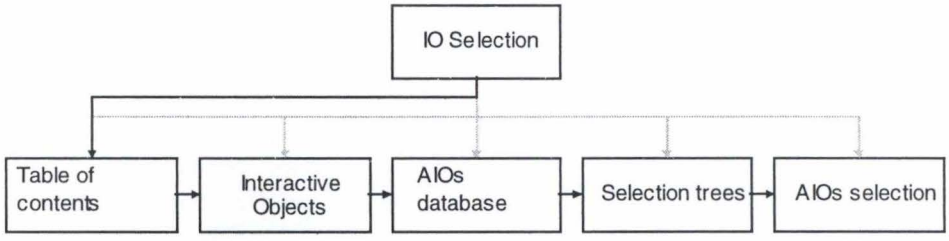


1.1.5. AIOs Selection Section

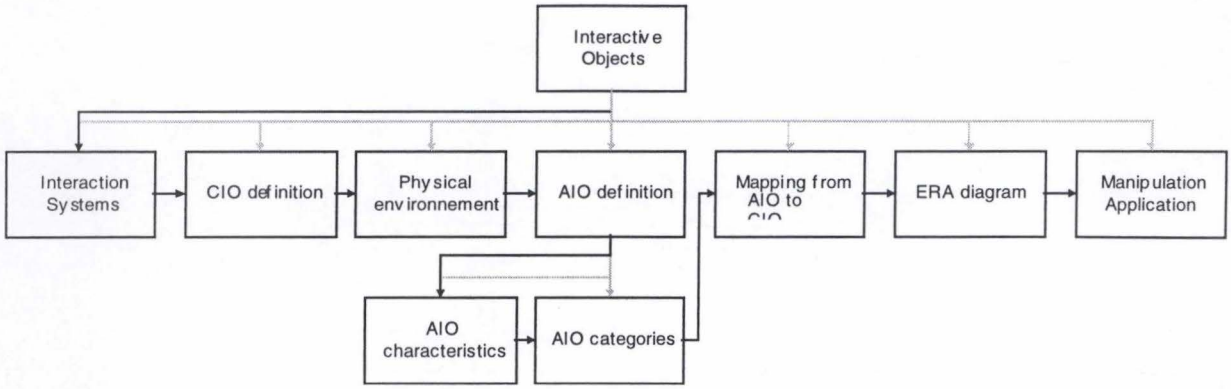


1.2. Sequential Navigation

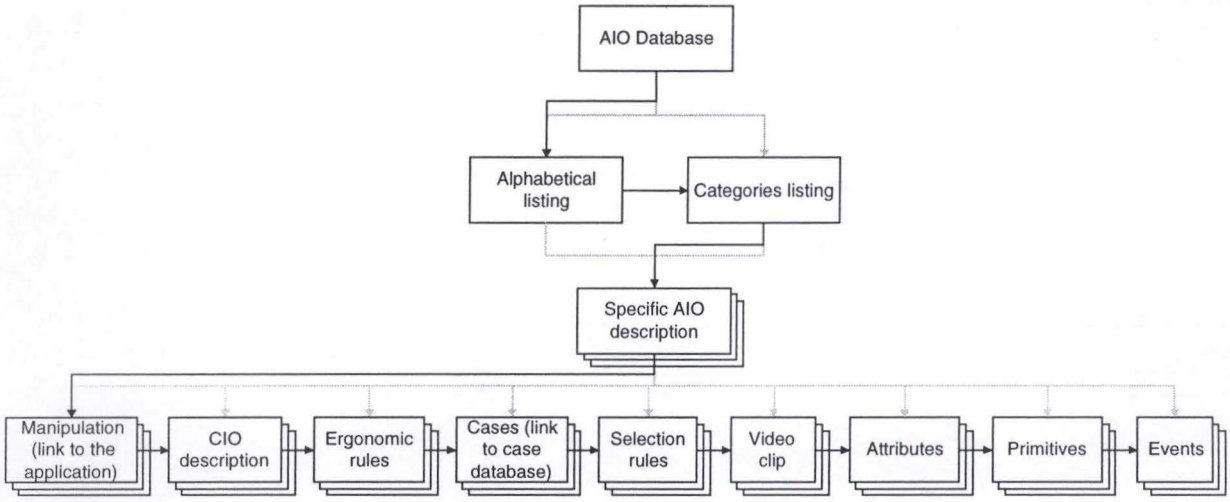
1.2.1. Top Level



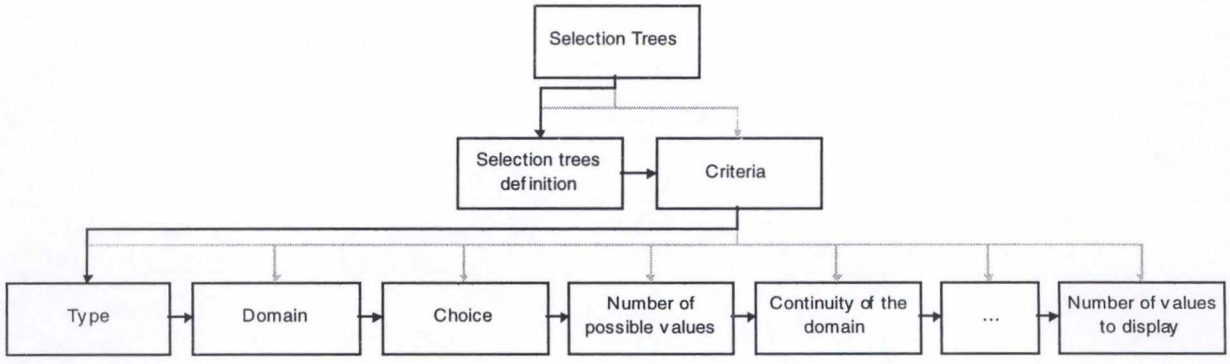
1.2.2. Interactive Objects Section



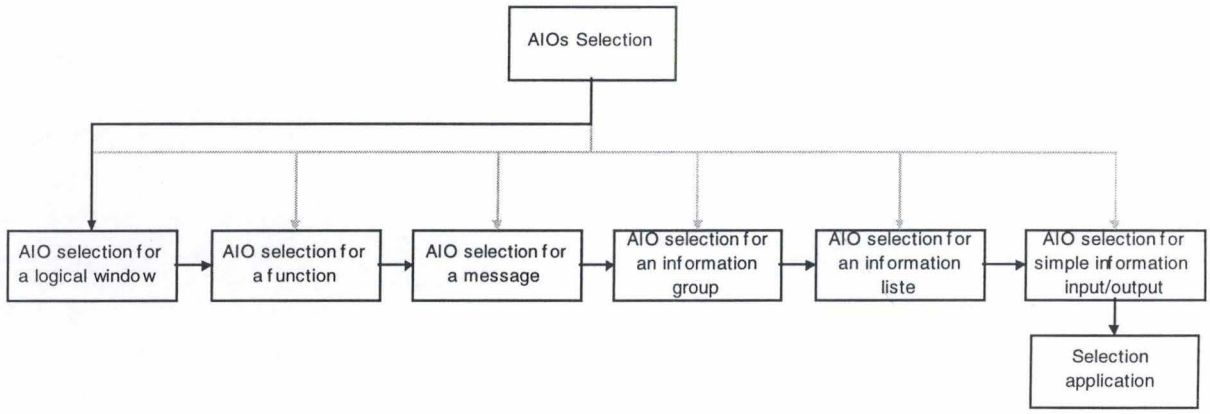
1.2.3. AIOs Database Section



1.2.4. Selection Trees Section



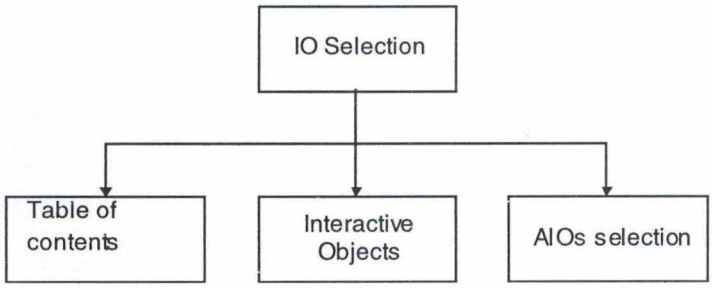
1.2.5. AIOs Selection Section



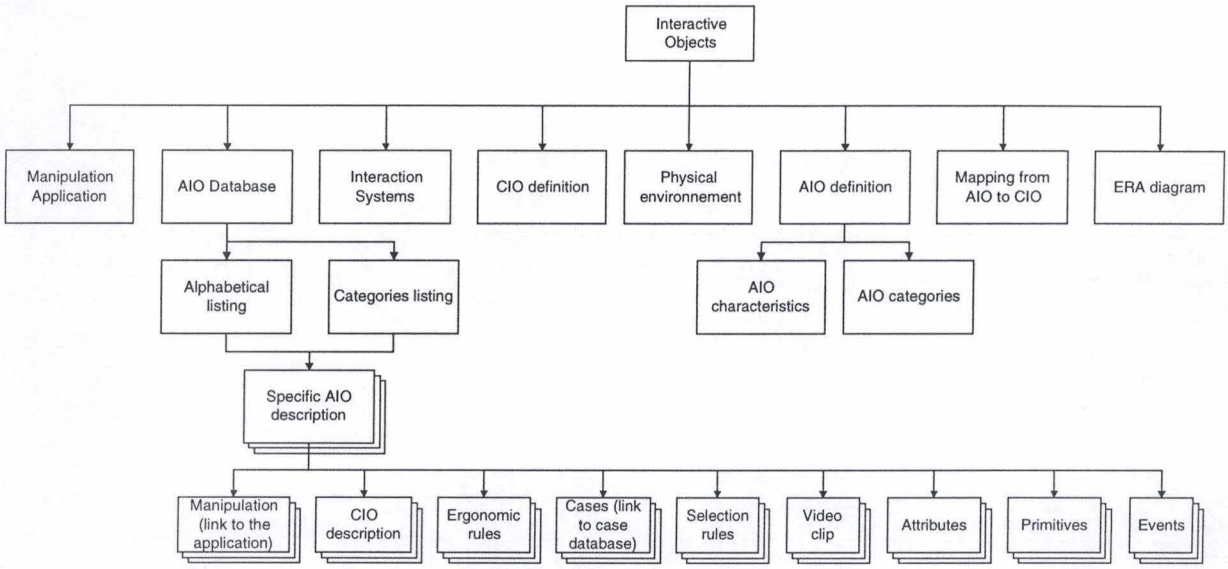
2. Constructivist Scenario

2.1. Hierarchical Navigation

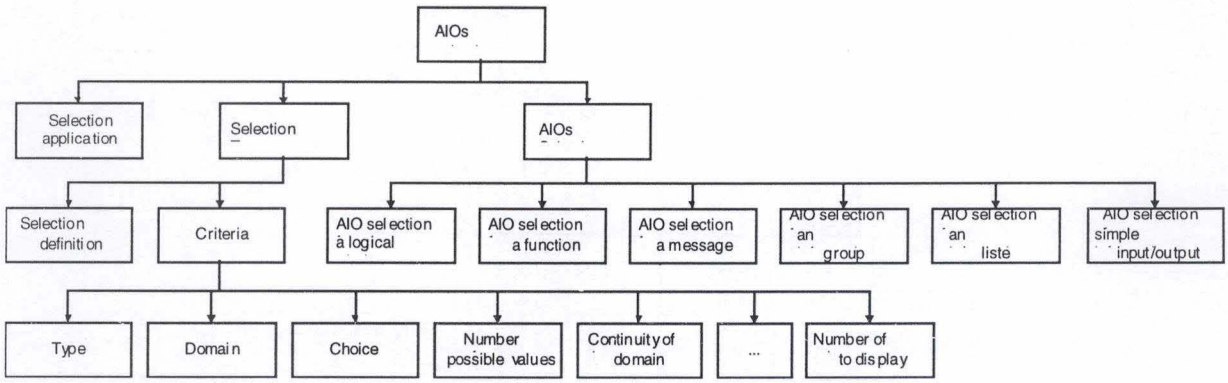
2.1.1. Top Level



2.1.2. Interactive Objects Section

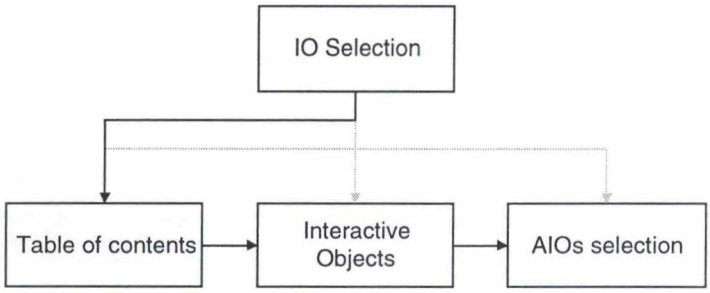


2.1.3. AIOs Selection Section

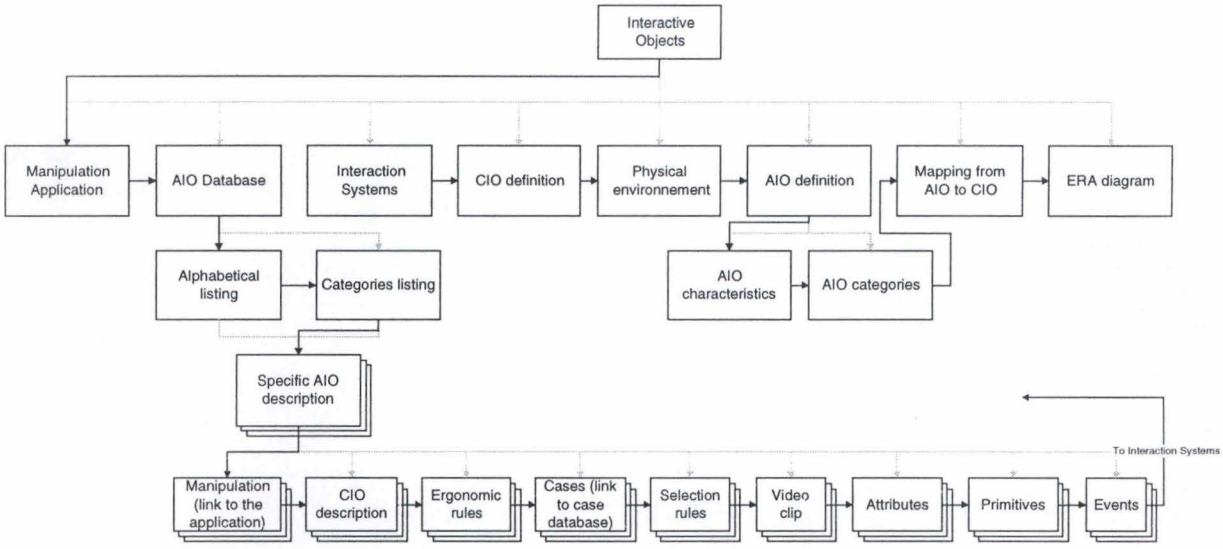


2.2. Sequential Navigation

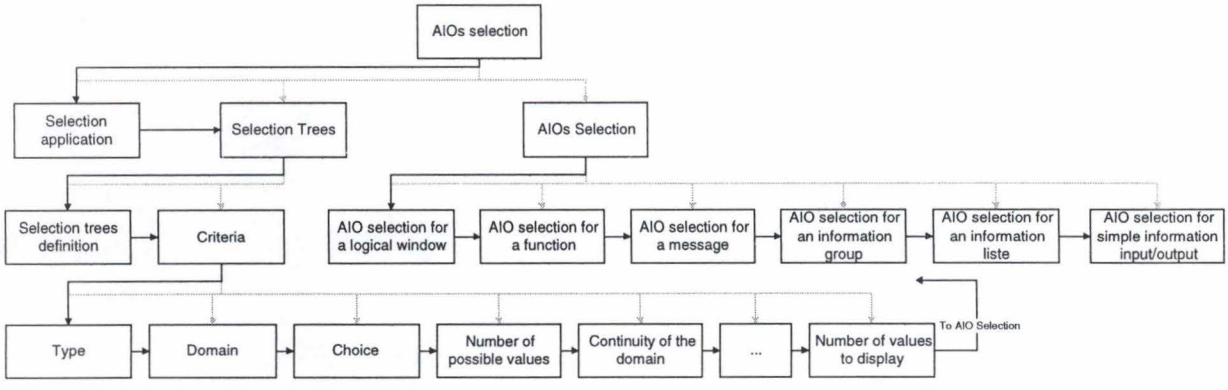
2.2.1. Top Level



2.2.2. Interactive Objects Section



2.2.3. AIOs Selection Section



D

Page Design

This section contains documents concerning the design of the pages.

1. Page Code

Here is the HTML code of a page template:

```
<html>
<head>
<title><VESALE=TITREPAGE></title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<LINK REL=STYLESHEET TYPE="text/css" HREF="style.css">
</head>

<body bgcolor="#FFFFFF">
<table border="0" CELLSPACING="0" CELLPADDING="0">
 <tr>
  <td rowspan="4" width="157" align="left" valign="top" bgcolor="#63639C">
   <table border="0" width="157" CELLSPACING="0" CELLPADDING="0">
    <tr>
     <td colspan="2"><img src="../images/logo.gif" width="157" height="116"></td>
    </tr>
    <tr>
     <td width="143" height="59" bgcolor="#63639C" valign="top"><img src="../images/pixel.gif" width="143"
height="1"><br><VESALE=RECH></td>
     <td width="14" height="59"><img src="../images/logomid.gif" width="14" height="59"></td>
    </tr>
    <tr>
     <td colspan="2"><img src="../images/logobas.gif" width="157" height="14"></td>
    </tr>
    <tr>
     <td colspan="2" bgcolor="#9CCE9C" valign="top"><VESALE=CHAPN>

                                          </td>
    </tr>
   </table>
   <p><VESALE=MENU></p>
   <p align="center"><img src="../images/line.gif" width="100" height="4"></p>
   <p> </p>

  </td>
  <td width="436" height="115" valign="top" bgcolor="#9CCE9C">
   <table width="436" height="115" border="0" CELLSPACING="0" CELLPADDING="0">
    <tr>
     <td width="436" height="14"><img src="../images/topbackground.gif" width="436" height="14"></td>
    </tr>
    <tr>
     <td width="436" height="79" valign="top"><VESALE=CHAPTER><VESALE=TITRE>

      </td>
    </tr>
    <tr>
     <td width="436" height="22" bgcolor="#FFFFFF"><img src="../images/round.gif" width="23" height="22"></td>
    </tr>
   </table>
  </td>
  <td height="115" width="157" valign="top" align="left">
   <table border="0" CELLSPACING="0" CELLPADDING="0">
    <tr>
     <td width="23" height="115" rowspan="3" valign="top"><img src="../images/prev-next-left.gif" width="23" height="115"></td>
     <td width="11" height="74" bgcolor="#63639C"><font size="1" color="#63639C">.</font><VESALE=FL></td>
     <td width="23" height="115" rowspan="3" valign="top"><img src="../images/prev-next-right.gif" width="23" height="115"></td>
    </tr>
    <tr>
     <td width="111" height="6"><img src="../images/prev-next-mid.gif" width="111" height="6"></td>
    </tr>
    <tr>
     <td width="111" height="35" bgcolor="#9CCE9C">
      <p CLASS="num" align="center"><font size="1" color="#9CCE9C">.</font><VESALE=NUM></p>
     </td>
    </tr>
   </table>
  </td>
 </tr>
 <tr>
```

```
  <td width="593" height="200" colspan="2" valign="top">
   <p> </p>
   </td>
 </tr>
 <tr>
  <td colspan="2"><img src="../images/navbottom.gif" width="593" height="22"></td>
 </tr>
 <tr>
  <td colspan="2" bgcolor="#63639C" valign="top">
   <table border="0" CELLSPACING="0" CELLPADDING="0" width="583" >
    <tr>
     <td width="511">
      <p CLASS="light" align="center"><font size="1" color="#63639C">.</font><VESALE=PIED></p>
     </td>
     <td width="72"><VESALE=FL>

     </td>
    </tr>
   </table>

  </td>
 </tr>
</table>
</body>
</html>
```

2. Style Sheet Code

Here is the style sheet code that we included in all our web pages:

```
p.menuchap { font-family: Arial, Helvetica, sans-serif; font-size: 10pt; font-weight: bold; color: #000000 ; text-transform: none}

p.chaph { font-family: Arial, Helvetica, sans-serif; font-size: 12pt; font-weight: bold; color: #63639C; text-transform: none}

p.titleh { font-family: Arial, Helvetica, sans-serif; font-size: 20pt; font-weight: bold; color: #000000; text-transform: none; text-indent: 5pt;
line-height: 3pt}

p.num { font-family: Arial, Helvetica, sans-serif; font-size: 14pt; font-weight: bolder; color: #63639C; text-transform: none}

p.light { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10pt; font-weight: bold; color: #8989B4; text-transform: none}

a.niv2 {  font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 8pt; font-weight: bold; text-indent: 15px; text-decoration: none; text-
align: left; clip:  rect(  ); color: #000000; height: 19px; width: 157px; line-height: 16px; vertical-align: middle}

a.niv3 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 8pt; font-weight: bold; text-indent: 15px; text-decoration: none; text-
align: left; clip: rect( ); color: #FFFFFF; height: 19px; width: 157px; line-height: 5pt; vertical-align: middle }

a.niv4 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 8pt; font-weight: bold; text-indent: 25px; text-decoration: none; text-
align: left; clip: rect( ); color: #FFFFFF; height: 19px; width: 157px; line-height: 5pt; vertical-align: middle }

a.sniv2 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 8pt; font-weight: bold; text-indent: 15px; text-decoration: none; text-
align: left; clip: rect( ); color: #000000; height: 19px; width: 157px; line-height: 16px; vertical-align: middle }

p.niv2 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 8pt; font-weight: bold; text-indent: 15px; text-decoration: none; text-
align: left; clip: rect( ); color: #000000; height: 19px; width: 157px; line-height: 16px; vertical-align: middle }

p.niv3 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 8pt; font-weight: bold; text-indent: 15px; text-decoration: none; text-
align: left; clip: rect( ); color: #9CCE9C; height: 19px; width: 157px; line-height: 5pt; vertical-align: middle }

a.chaph { font-family: Arial, Helvetica, sans-serif; font-size: 12pt; font-weight: bold; color: #63639C; text-transform: none ; text-decoration:
none; clip:  rect(  ); text-indent: 5px; border-color: #9CCE9C black black}

h1 {  font-family: "Comic Sans MS"; font-size: 18pt; text-decoration: underline; font-style: italic; text-transform: none}

h2 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 12pt; color: #414167; text-decoration: underline ; text-indent: 10pt}
td.core {  margin-left: 3px}

a.chapm { font-family: Arial, Helvetica, sans-serif; font-size: 10pt; font-weight: bold; color: #63639C; text-transform: none ; text-decoration:
none; clip: rect( ); border-color: #9CCE9C black black ; border: black}

.fig {  font-family: Arial, Helvetica, sans-serif; font-size: 9pt; font-style: italic}
```

```
h2 {  font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 16pt; color: #006600; margin-left: 10px; font-weight: lighter}

p.titre {  font-weight: 700; font-family: Verdana, Arial, Helvetica, sans-serif}

p.texte {  font-family: Georgia, "Times New Roman", Times, serif; font-size: 12pt; margin-left: 5px}
```

E

Dynamic Generation

This section contains documents and code related to the dynamic generation of the IOs descriptions pages as well as as to the dynamic navigation.

1. Course structure syntax

Here is the syntax of the course structure description:

```
<structure> = <startchar> <page>* <endchar>

<startchar> = "D"

<endchar> = "F"

<linestartchar> = "*"

<lineendchar> = "*"

<separator> = "#"


<page> = <linestartchar> <level> <separator> <shortname>
<separator> <longname> <separator> <url> <lineendchar>

<level> = 1 | 2 | 3 | 4| 5

<shortname> = chaine de caractère

<longname> = chaine de caractère

<url> = chaine de caractère
```

2. Pages examples

This section contains a few page examples.

2.1. Page containing the <VESALE> tags

The figure E-1 illustrates a page containing the VESALE tags that will have to be replaced by the HTML code implementing the navigation elements.

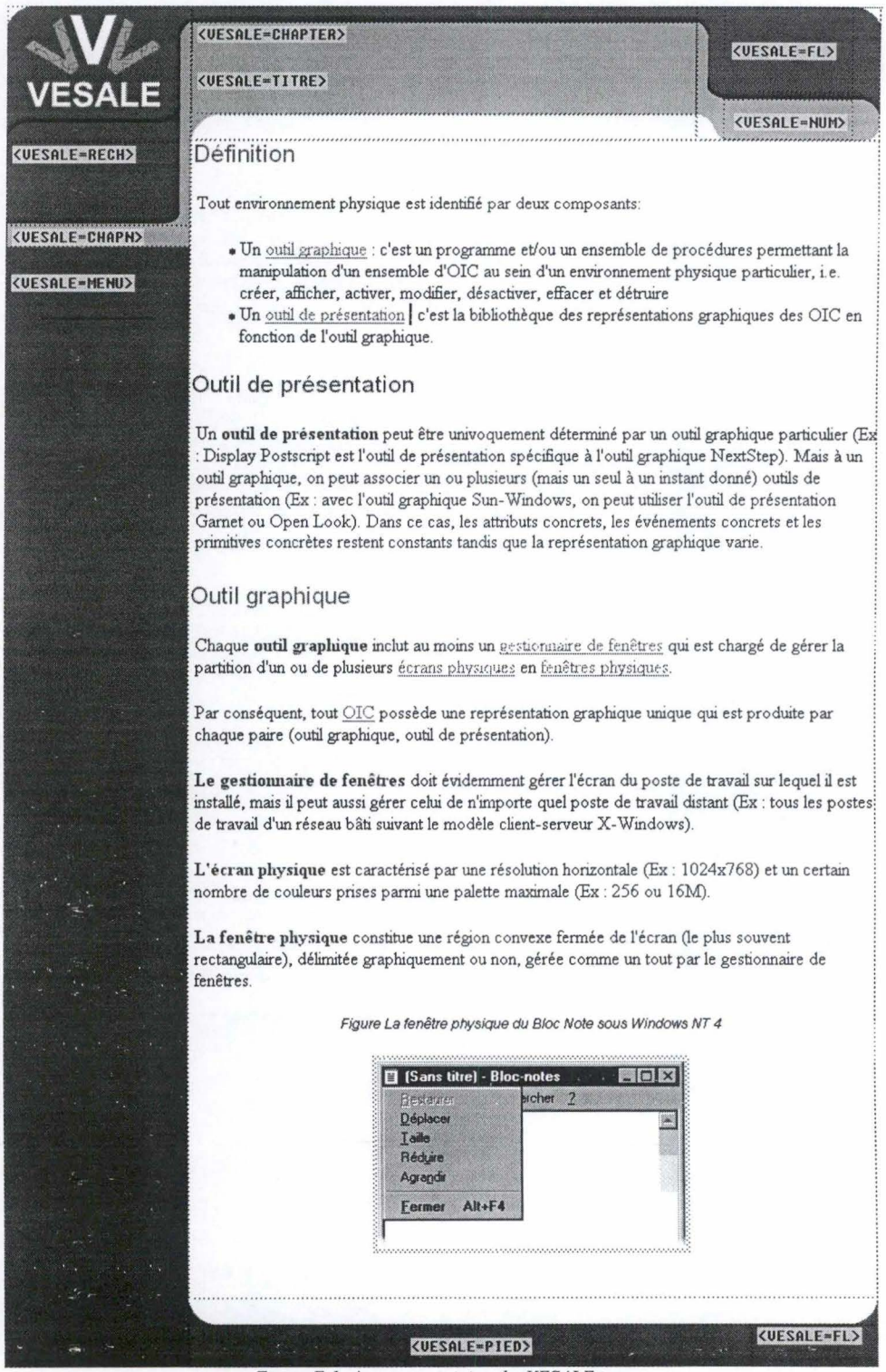


Figure E-1. A page containing the VESALE tags

2.2. Page in which the <VESALE> tags have been replaced

The figure E-2 shows the page displayed in figure E-1 in which the VESALE tags have been replaced.

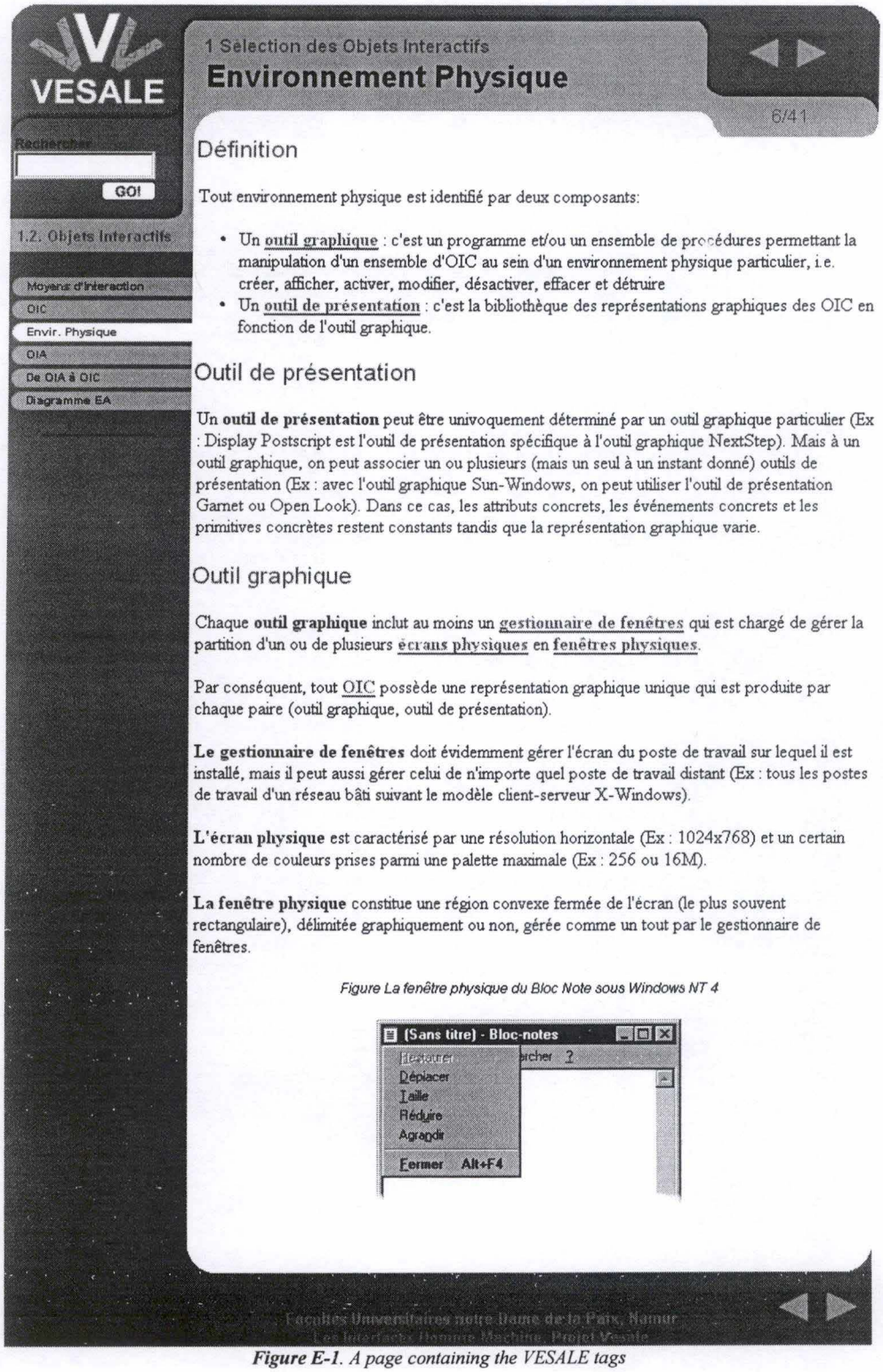


Figure E-1. *A page containing the VESALE tags*

3. Source Code

3.1. Navigation

```
package Vesale.Navigation;

import java.io.*;
import java.util.*;
import java.awt.event.*;
import java.lang.*;

//                                                              //
//                    Navigation                                //
//                                                              //

public class Navigation {

//                                                              //
//                    Constants                                 //
//                                                              //

protected static int STARTLETTER = 'D';
protected static int ENDLETTER = 'F';
protected static int FIELDSEPARATOR = '#';
protected static char EMPTYCHAR = '-';
protected static char RULESTART = '*';
protected static char q = '"';

//                                                              //
//                    Internal variables                        //
//                                                              //

private Vector navTree = new Vector();
private int[] curniv = {0,0,0,0,0};
private int actniv=0;
private File inputFile = new File("navstruct.txt");
private FileReader in;
private char c;
private int currentPos=0;
private String currentURL = "";

//                                                              //
//                    Initialization                            //
//                                                              //

public Navigation ()
{
  parseNavText();
}

//                                                              //
//                    Private methods                           //
//                                                              //

/** readoneline
* this method read a line in the
* tree text file **/
```

```
public void readoneline () throws IOException
{...}


/** parseNavText
* this method build the navigation tree
* based on a formated text file**/

public void parseNavText ()
{...}


/** searchPos
* this method returns the position of the page
* identified by its URL in the navTree by setting
* the global variables currentPos and currentURL
* to the right value
**/

private void searchPos (String lurl)
{
  int i=0;
  boolean found = false;
  if (lurl!=currentURL)
  {
    while (i<navTree.size())
    {
      MenuItem men = (MenuItem)navTree.elementAt(i);
      String curl = men.getPage().getURL();
      if (curl.compareTo(lurl)==0)
      {
        found=true;
        currentURL=lurl;
        currentPos=i;
        currentURL=lurl;
        i=navTree.size();
      }
      else {i++;};
    };
  }
  else {found=true;};
  if (found==false)
  {
    currentPos=0;
    currentURL="Not Found";
  };
}


/** getTitle
* This methods returns the PageTitle related to
* the given URL. A PageTitle is composed of the
* chapter's name, its index and URL and the
* page title.**/

private PageTitle getTitle (String urlReq)
{...}


/** getArrowZone
* This methods returns the ArrowZone related to
```

```
* the given URL. An ArrowZone is composed of the
* level of the current page in the hierarchy,
* the next and previous <Page>, the position of
* the page in the hierarchy and the total number
* of pages in the course. A <Page> is composed of
* a page name and a URL.**/

private ArrowZone getArrowZone (String lurl)
{...}


/** getMenu
* This methods returns the menu related to
* the given URL. A menu is a Vector of <MenuItem>
* A menuItem is composed of a <Page> and
* a index value**/

private Vector getMenu (String lurl)
{
  searchPos(lurl);
  Vector vmenu = new Vector();
  MenuItem mi = (MenuItem)navTree.elementAt(currentPos);
  int [] curIndex = mi.getIndex();
  int curniv = mi.getLevel();
  int [] tstIndex;
  MenuItem test;
  for (int j=0; j<navTree.size();j++)
  {
    test = (MenuItem)navTree.elementAt(j);
    tstIndex = test.getIndex();

    if (curniv==1) {if
    ((tstIndex[0]==curIndex[0])&&(tstIndex[1]!=0)&&(tstIndex[2]==0))
    {vmenu.addElement(test);};};

    if (curniv>=2) {if
    ((tstIndex[0]==curIndex[0])&&(tstIndex[1]==curIndex[1])&&
    (tstIndex[2]!=0)&&(tstIndex[3]==0)) {vmenu.addElement(test);};};

    if (curniv>=3) {if
    ((tstIndex[0]==curIndex[0])&&(tstIndex[1]==curIndex[1])&&
    (tstIndex[2]==curIndex[2])&&(tstIndex[3]!=0)&&(tstIndex[4]==0))
    {vmenu.addElement(test);};};

    if (curniv>=4) {if
    ((tstIndex[0]==curIndex[0])&&(tstIndex[1]==curIndex[1])&&
    (tstIndex[2]==curIndex[2])&&(tstIndex[3]==curIndex[3])&&
    (tstIndex[4]!=0)&&(tstIndex[5]==0)) {vmenu.addElement(test);};};
    };
  return vmenu;
}


/** getIndex
* This methods returns the table of content of
* the chapter that includes the given URL.
* A table of content is a Vector of <MenuItem>**/

private Vector getIndex (String lurl)
{...}
```

```
/** getStructure
* This methods returns the section's structure
* related to the given URL. A strcture is composed
* of the section's name, its index and the URL of
* the section's main page
**/

private Structure getStructure (String lurl)
{...}

//                                                    //
//                  Public methods                    //
//                                                    //

/** getTitleHTML
* This methods returns the HTML code
* of a title for the specified URL
**/

public String getTitleHTML (String lurl)
{...}


/** getPageTitleHTML
* This methods returns the HTML code
* of a page title for the specified URL
**/

public String getPageTitleHTML (String lurl)
{...}


/** getChapterHTML
* This methods returns the HTML code
* of a chapter title for the specified URL
**/

public String getChapterHTML (String lurl)
{...}


/** getNumPagesHTML
* This methods returns the HTML code
* for the position of the page specified
* by its URL in the total number of pages
* in the course.
**/

public String getNumPagesHTML (String lurl)
{...}


/** getArrowZoneHTML
* This methods returns the HTML code
* of the arrow zone for the specified URL
**/

public String getArrowZoneHTML (String lurl)
{...}
```

```
/** getMenuHTML
* This methods returns the HTML code
* of a menu for the specified URL
**/

public String getMenuHTML (String lurl)
{
try
{
  String code = ("");
  Vector vmenu = getMenu(lurl);
  int[] curniv = ((MenuItem)navTree.elementAt(currentPos)).getIndex();
  MenuItem mi;
  int[] miniv;
  for (int i=0;i<vmenu.size();i++)
  {
    mi = (MenuItem)vmenu.elementAt(i);
    miniv = mi.getIndex();
    if (mi.getLevel()==2) {code=
      code+"\n<table   border="+q+"0"+q+"   width="+q+"157"+q+"   CELLSPAC-
      ING="+q+"0"+q+"             CELLPADDING="+q+"0"+q+"              back-
      ground="+q+"../images/back.gif"+q+">\n<tr>\n<td>
      <a CLASS="+q+"niv2"+q+" href="+q+".."+mi.getPage().getURL()+q+">"
      +mi.getPage().getShortname()+"</a></td>\n</tr>\n</table>\n";};
    if (mi.getLevel()==3)
    {
      if ((miniv[1]==curniv[1])&&(miniv[2]==curniv[2]))
      {
        code="...";
      }
      else
      {
        code="...";
      };
    };
    if (mi.getLevel()>=4) {
      if (miniv==curniv)
      {
        code="...";
      }
      else
      {
        code="...";
      };
    };
  };
  return code;
}
catch (Exception e) {return e.toString();};
}

/** getIndexHTML
* This methods returns the HTML code
* of a table of content for the
* specified URL
**/

public String getIndexHTML (String lurl)
{...}
```

```
/** getStructureHTML
* This methods returns the HTML code
* of the section's structure for
* the specified URL
**/

public String getStructureHTML(String lurl)
{...}


/** getFooterHTML
* This methods returns the HTML code
* of the page footer
**/

public String getFooterHTML()
{...}


/** getSearchHTML
* This methods returns the HTML code
* of the search engine (a gif image at
* the moment)
**/

public String getSearchHTML(String lurl)
{...}
```

3.2. Replacer

```
package Vesale.Tools;

import java.util.*;
import java.sql.*;
import java.awt.event.*;
import Vesale.DB.*;
import java.lang.*;
import java.io.*;

public class Replacer
{

 Vector replacements = new Vector();
 public String firstTag = "VESALE";

/* empty constructor */
 public Replacer() {}

/* add a replacement */
public void addReplacement (ReplacedTag tag)
{
  replacements.addElement(tag);
}

/* clear all the replacements */
public void clearReplacements()
{
  replacements.clear();
}
```

```
/* replace all the <firsttag=value> strings with the specified value */
public String replaceLine(String line)
{
  try
  {
    int index = 0;
    String searchedString = "<" + firstTag;
    String tag = new String();
    String newString = new String();
    while ((index = line.indexOf(searchedString,index)) >= 0)
    {
      /* index + firstTag.length() + 1 = indice du premier caractère de
      la valeur à remplacer */
      tag    =    line.substring(index    +    firstTag.length()    +    2,
line.indexOf(">",index + 1));
      newString = getNewString(tag);
      if (newString.length() >= 0 )
      {
        line = line.substring(0,index) + newString + line.substring(index +
3 + tag.length() + firstTag.length());
        index += newString.length();
      }
    }
    return line;
  }
  catch (NullPointerException e) {return new String ("NullPointer dans re-
place");}
}

/* returns the string that will replace a specific tag*/
private String getNewString(String tag)
{
  try
  {
    for (int i = 0; (i < replacements.size()) ; i++)
    {
      if (((ReplacedTag)replacements.elementAt(i)).tag.compareTo(tag) == 0)
return  ((ReplacedTag)replacements.elementAt(i)).value;
    }
    return new String();
  }
  catch (NullPointerException e) { return new String("NullPointerException
replace: " + e.toString());}
}
}

package Vesale.Tools;

import java.util.*;
import java.sql.*;
import javax.swing.*;
import java.awt.event.*;
import java.lang.*;
import java.io.*;

public class ReplacedTag
{

public String tag = new String();
public String value = new String();
```

```
public ReplacedTag(String tag, String value)
{
  this.tag = tag;
  this.value = value;
}

public ReplacedTag()
{
}


public String toString()
{
  return "(" + tag + " par " + value + ")";
}
}
```

3.3. IODBReader

```
package Vesale.DB;

import java.util.*;
import java.sql.*;
import java.awt.event.*;
import java.lang.*;
import java.io.*;
import Vesale.Tools.*;

public class AIODBReader
{
  Connection conn = null;
  int count = 0;
  /* Prepared statements */
  PreparedStatement listAttributes;
  PreparedStatement attributeDescription;
  PreparedStatement infoAttribute;
  /* event */
  PreparedStatement listEvents;
  PreparedStatement eventDescription;
  /* primitive */
  PreparedStatement listPrimitives;
  PreparedStatement primitiveDescription;
  /* AIO */
  PreparedStatement infoAIO;
  PreparedStatement getAIOName;
  PreparedStatement listAIOs;
  PreparedStatement listEventsAIO;
  PreparedStatement listAttributesAIO;
  PreparedStatement listPrimitivesAIO;
  PreparedStatement listIsInheritedByAIO;
  PreparedStatement listInheritsFromAIO;
  /* CIO*/
  PreparedStatement getCIOs;
  PreparedStatement getAIOPic;
  /* rules */
  PreparedStatement getRules;


/* Constructor */
public AIODBReader()
```

```
 {
   try
   {
     DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver ());
     debug("Driver OK");
   }
   catch (Exception e) {debug("Driver KO"); }
   connect();
 }


/* disconnect */
public void disconnect()
{
  try
  {
    conn.close();
  }
  catch (Exception e) {}
}


/* connect */
private void connect()
{
  try
  {
    conn                    =                    DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:orc3" ,"scott" , "tiger");
     debug("Connexion OK");
   }
  catch (SQLException s) {debug("Connexion KO");}
  try
  {
/* sql prepared statements */
/* here follows the initialisation of all the sql queries used in this
class */
/* attributes */
    listAttributes = conn.prepareStatement("select name from attribute or-
der by name ");
    infoAttribute = conn.prepareStatement("select description from attrrib-
ute where name = ?");
      attributeDescription = conn.prepareStatement("select description from
attribute where name = ?");
      /* events */
      listEvents = conn.prepareStatement("select name from event order by
name ");
      eventDescription = conn.prepareStatement("select description from
event where name = ?");
      /* primitives */
      listPrimitives = conn.prepareStatement("select name from primitive
order by name ");
      primitiveDescription = conn.prepareStatement("select description from
primitive where name = ?");
      /* aio */
      listAIOs = conn.prepareStatement("select frname, idaio from aio order
by frname ");
      infoAIO = conn.prepareStatement("select enname, frname, definition,
category, manipurl from aio where idaio = ? ");
```

```
      listAttributesAIO = conn.prepareStatement("select attribute.name, at-
tribute.description from aioattr, attribute where idaio = ? and attrib-
ute.name = aioattr.name");
     listEventsAIO      =      conn.prepareStatement("select      event.name,
event.description from aioevent, event where idaio = ? and event.name =
aioevent.name");
      listPrimitivesAIO  =  conn.prepareStatement("select  primitive.name,
primitive.description from aioprim, primitive where idaio = ? and primi-
tive.name = aioprim.name");
      listInheritsFromAIO  =  conn.prepareStatement("select    idaio,frname
from aio where idaio in(select distinct inheritance.inh_idaio from inheri-
tance where idaio = ?)");
      listIsInheritedByAIO  =  conn.prepareStatement("select    idaio,frname
from aio where idaio in (select distinct inheritance.idaio from inheritance
where inh_idaio = ?)");
      getAIOName  =  conn.prepareStatement("select   frname  from  aio  where
idaio = ?");
      getAIOPic  =  conn.prepareStatement("select  representation  from  cio
where idaio = ?");
      /* cio */
      getCIOs      =      conn.prepareStatement("select      idaio,      id-
cio,name,description,graphtool,prestool,  representation  from  cio  where
idaio = ?");
      /* rule */
      getRules  =  conn.prepareStatement("select  name,  rule,  posex,  negex
from ergrule where idrule in (select idrule from aiorule where idaio =
?)");
      debug("Statement OK");
    }
    catch (SQLException s) {debug("Statement KO");}
 }


public ErgRule[] getRelatedRules(String idaio)
{
  try
  {
      Vector vec = new Vector();
      synchronized(this)
      {
       getRules.clearParameters();
       getRules.setString(1,idaio);
       ResultSet res = getRules.executeQuery();
       while (res.next())
       {
          ErgRule ergrule = new ErgRule();
          ergrule.setName(res.getString(1));
          ergrule.setEnonce(res.getString(2));
          ergrule.setPosex(res.getString(3));
          ergrule.setNegex(res.getString(4));
          ergrule.setPosexillustr(res.getString(5));
          ergrule.setNegexillustr(res.getString(6));
          ergrule.setJustification(res.getString(7));
          vec.addElement(ergrule);
         }
      }
      return ToolBox.ErgRuleVectorToArray(vec);
    }
    catch (SQLException s) { return new ErgRule[]  {};}
}
```

```
/*                                                    */
/* Opérations sur les OIC                             */
/*                                                    */

public CIO[] getRelatedCIOs(String idaio)
{
  try
  {
      Vector vec = new Vector();
      synchronized(this)
      {
       getCIOs.clearParameters();
       getCIOs.setString(1,idaio);
       ResultSet res = getCIOs.executeQuery();
       while (res.next())
       {
          CIO cio = new CIO();
          cio.setDescription(res.getString(4));
          cio.setIdaio(new AIO (idaio, getAIOName(idaio)));
          cio.setName(res.getString(3));
          cio.setGraphtool(res.getString(5));
          cio.setPrestool(res.getString(6));
          cio.setRepresentation(res.getString(7));
          vec.addElement(cio);
        }
      }
      return ToolBox.CIOVectorToArray(vec);
    }
    catch (SQLException s) { return new CIO[]  {};}
}

and so on.....

}
```

3.4. NavigationTagReplacer

```
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import java.math.*;
import javax.servlet.*;
import javax.servlet.http.*;
import Vesale.DB.*;
import Vesale.Tools.*;
import nav.Navigation.*;

public class TagReplacerServlet extends HttpServlet
{

  ToolBox t = new ToolBox();
  Navigation nav;
  AIODBReader db;

  /* initialisation */
  public void init (ServletConfig config) throws ServletException
  {
    super.init(config);
    try
```

```
    {
      db = new AIODBReader();
      log("Connect");
      nav = new Navigation();
      log("Nav created");
    }
    catch (Exception e) { log("db not connected: " + e.toString()); }
  }


 /* destroy method */
 /* execute when the servlet is freed from the memory */
 public void destroy()
 {
   db.disconnect();
   log("Disconnect");
 }


/* DoGet Method : this method is executed when the servlet receives a GET
HTTP request */
public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
  {
     ServletOutputStream out = res.getOutputStream();
     String file = req.getPathTranslated();
     if (file == null)
     {
      out.println("The request file doesn't exist");
      return;
     }
     res.setContentType("text/html");
     /* reads the requested file */
     String content = t.readFile(new File(file));
     /* generates a replacer object according to the requested URI */
     Replacer replacer = getReplacer(req.getRequestURI());
     /* adds replacements if the requested URL is a dynamic page */
     replacer = modifyReplacer(replacer,req);
     /* returns the modified file */
     out.println(replacer.replaceLine(content));
   }


/* adds replacements if the requested file is a dynamic page */
public Replacer modifyReplacer(Replacer replacer, HttpServletRequest req)
{
  try
  {
    String uri = removeParameters(req.getRequestURI());
    if (uri.compareTo("/oi/oia/ficheoia.html") == 0)
    {
      /* OIA */
      InfoAIO info = db.getInfoAIO(req.getParameter("oia"));
      replacer.addReplacement(new                                    Replaced-
Tag("IDAIO",req.getParameter("oia")));
      replacer.addReplacement(new                                    Replaced-
Tag("ENGNAME",info.getEngname()));
      replacer.addReplacement(new ReplacedTag("FRNAME",info.getFrname()));
      replacer.addReplacement(new                                    Replaced-
Tag("CATEGORIE",info.getCategory()));
```

```
      replacer.addReplacement(new                                    Replaced-
Tag("DEFINITION",info.getDefinition()));
      replacer.addReplacement(new                                    Replaced-
Tag("HERITEDE",HTMLConverter.AIOArrayToHTML(info.getInheritsfrom())));

      replacer.addReplacement(new    ReplacedTag("ESTHERITEPAR",    HTMLCon-
verter.AIOArrayToHTML(info.getIsinheritedby())));
      replacer.addReplacement(new                                    Replaced-
Tag("OIA",req.getParameter("oia")));
      if (info.getCIO() != null)
      {
        replacer.addReplacement(new     ReplacedTag("OUTIL",     "("     +
info.getCIO().getGraphtool() + ")"));
        replacer.addReplacement(new ReplacedTag("REPRESENTATION", HTMLCon-
verter.picToHTML(info.getCIO().getRepresentation(),
info.getCIO().getName())));
      }
      else
      {
        replacer.addReplacement(new ReplacedTag("OUTIL",""));
        replacer.addReplacement(new    ReplacedTag("REPRESENTATION","Aucune
représentation disponible"));
      }
    }
    else if (uri.compareTo("/oi/oia/ficheattributs.html") == 0)
    {
                and so on...
      }
      return replacer;
  }
  catch (Exception e) { log ("plante add replacer :" + e.toString()); re-
turn new Replacer();}
}

public String removeParameters(String url)
{
  try
  {
    return url.substring(0,url.indexOf("?"));
  }
  catch (Exception e) { return url;}
}


/* returns the replacer object initialised with the replacements for the
navigational elements */
public Replacer getReplacer(String uri)
{
  Replacer replacer = new Replacer();
  String file = removeParameters(uri);
  replacer.addReplacement(new                          ReplacedTag("CHAPTER",
nav.getChapterHTML(file)));
  replacer.addReplacement(new ReplacedTag("MENU", nav.getMenuHTML(file)));
  replacer.addReplacement(new ReplacedTag("PIED",nav.getFooterHTML()));
  replacer.addReplacement(new ReplacedTag("TITRE",nav.getTitleHTML(file)));
  replacer.addReplacement(new ReplacedTag("RECH",nav.getSearchHTML(file)));
  replacer.addReplacement(new                                    Replaced-
Tag("FL",nav.getArrowZoneHTML(file)));
  replacer.addReplacement(new                                    Replaced-
Tag("NUM",nav.getNumPagesHTML(file)));
```

```
  replacer.addReplacement(new                                    Replaced-
Tag("CHAPN",nav.getStructureHTML(file)));
  replacer.addReplacement(new                                    Replaced-
Tag("PAGETITLE",nav.getPageTitleHTML(file)));
  return replacer;
}
}
```

4. IOs description pages

This section contains screen captures of the IOs description pages.

The figure E-3 shows the AIO description page for the Edit Box object.

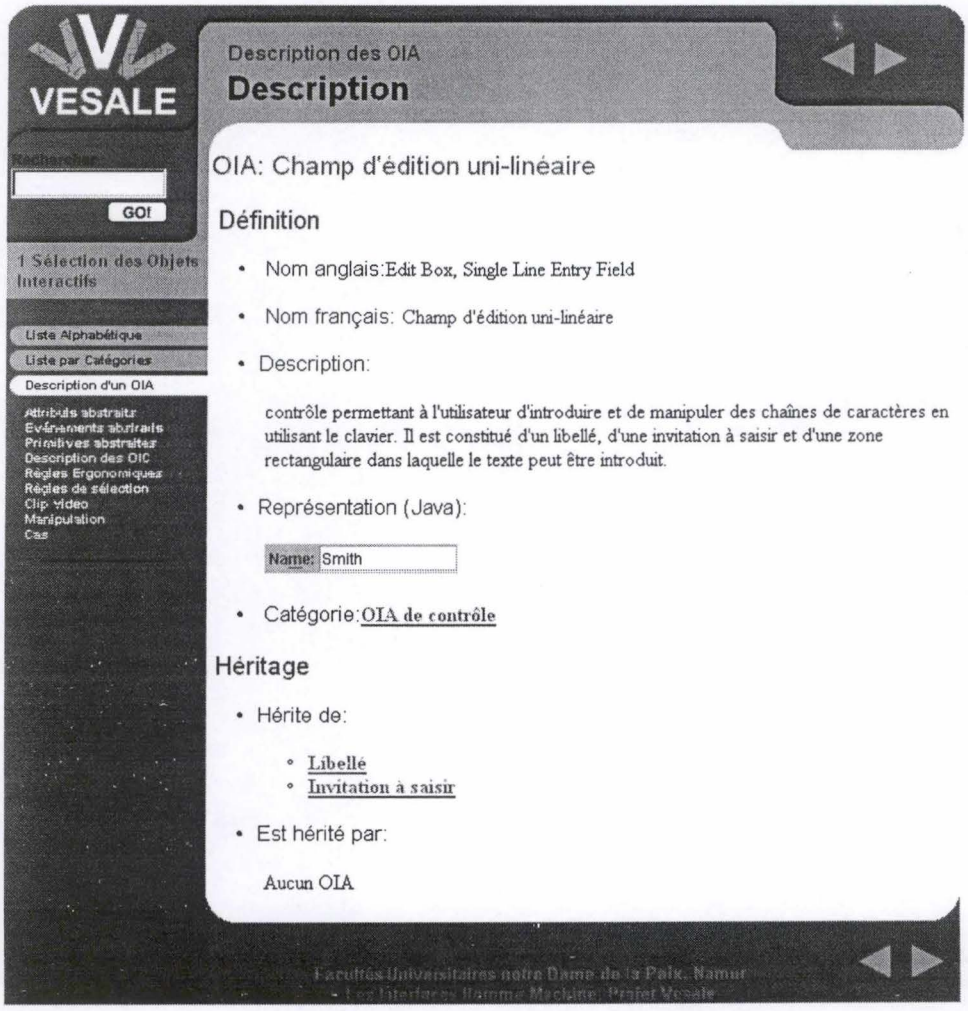


Figure E-3. The AIO description page

The figure E-3 shows the CIOs description page for the Edit Box object.

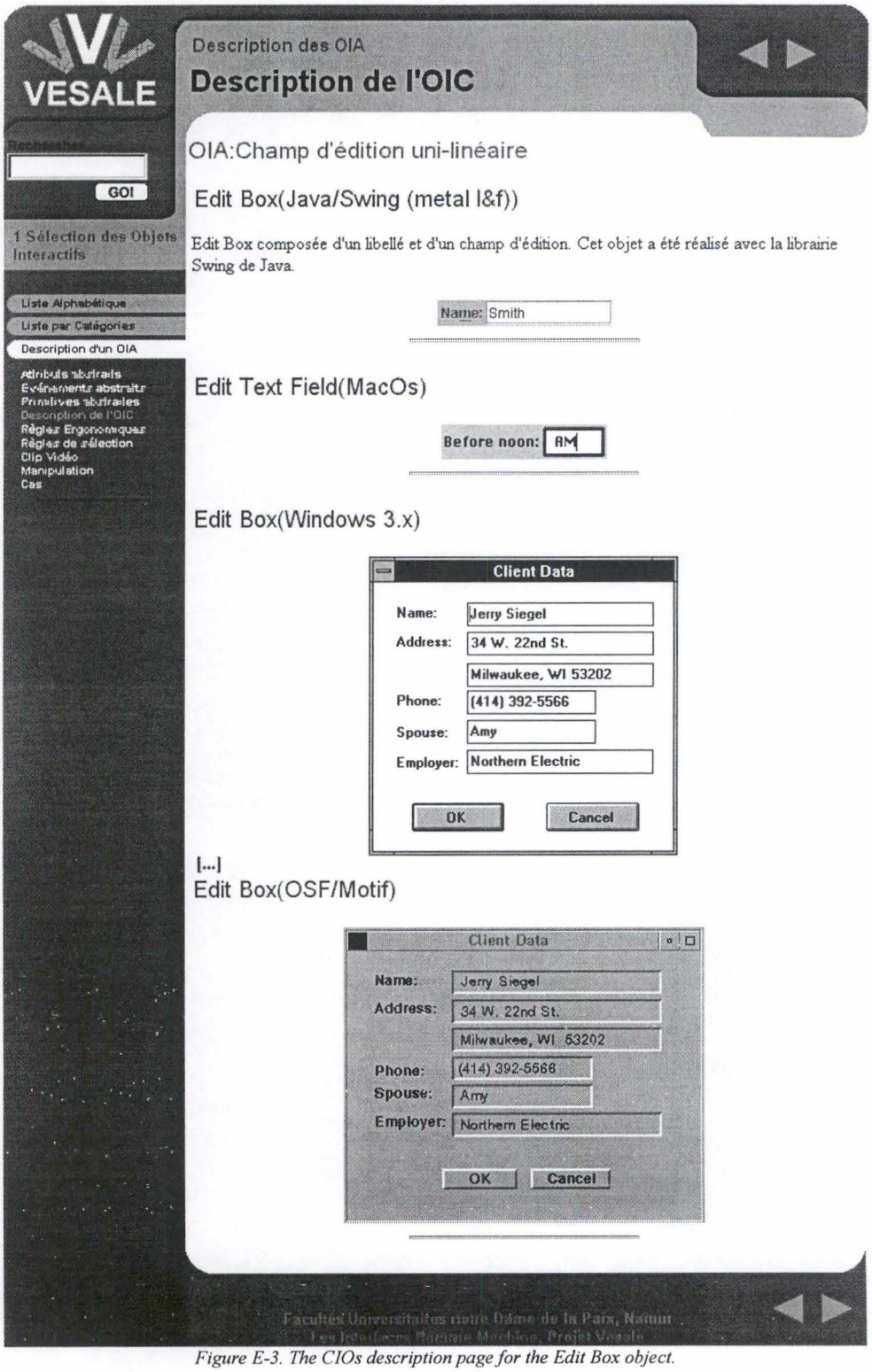


Figure E-3. The CIOs description page for the Edit Box object.

The figure E-4 shows the ergonomic rules description page for the Edit Box object.

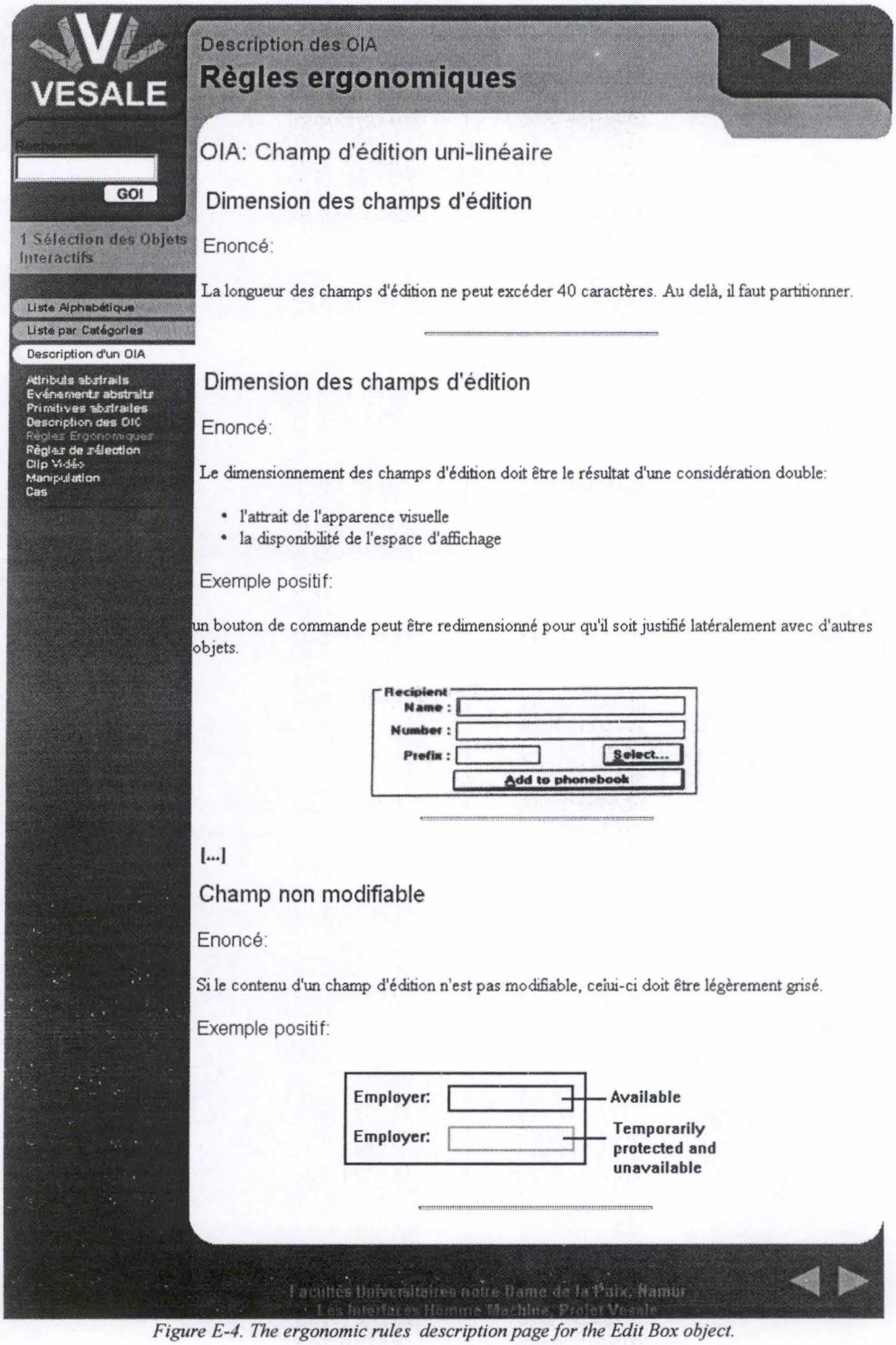


Figure E-4. The ergonomic rules description page for the Edit Box object.

The figure E-5 shows the attributes description page for the Edit Box object.

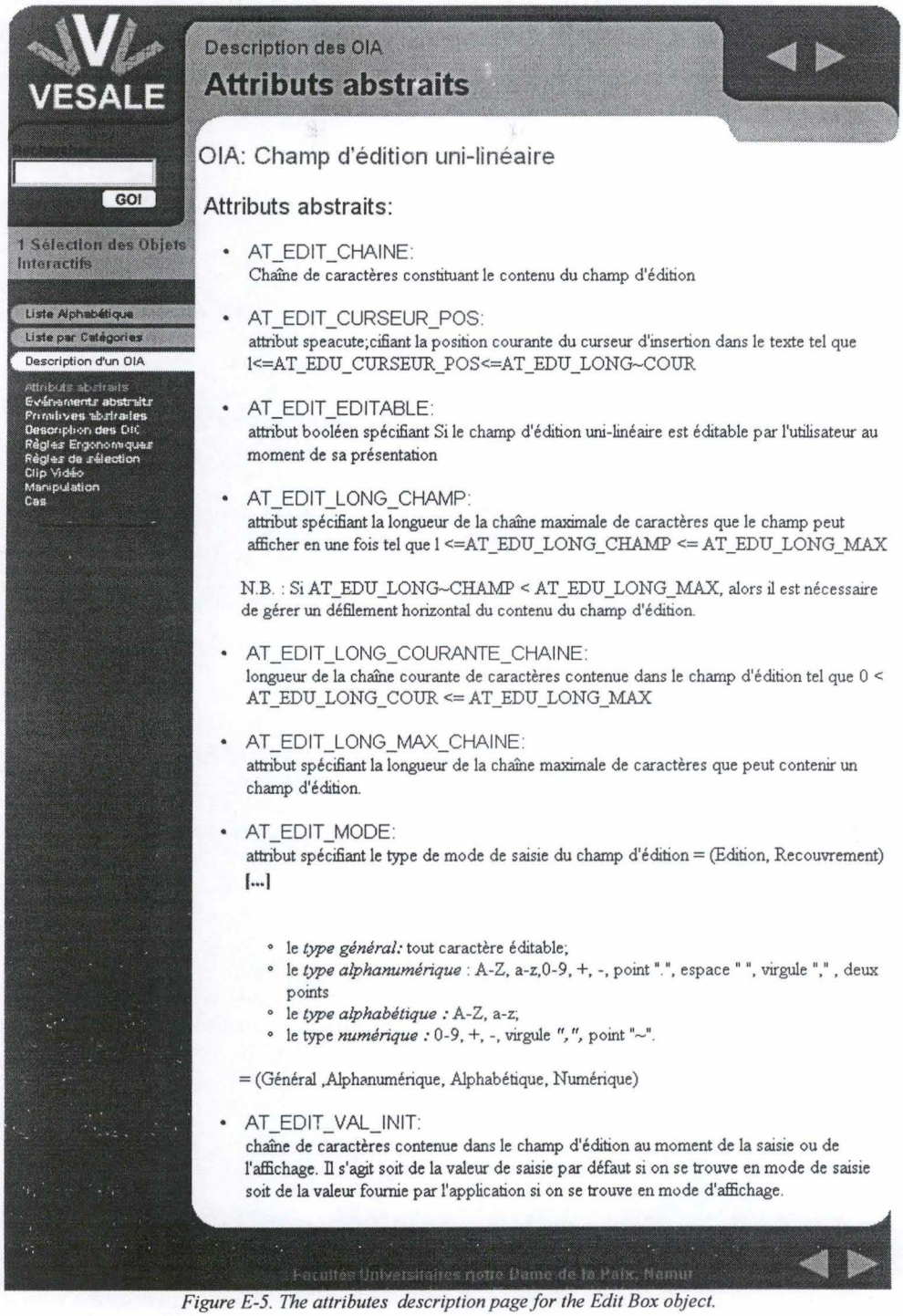


Figure E-5. The attributes description page for the Edit Box object.

5. Course structure definition

Here is the definition of the course structure:

```
D*1#Sélection des Objets Interactifs#Sélection des OI#oi/index.htm*
*2#Table des Matières#TDM#oi/tdm/tdmoi.html*
*2#Objets Interactifs#OI#oi/oi.html*
*3#Moyens d'Interaction#Moyens d'Interaction#/oi/oi/moyint.html*
*3#Objets Interactifs Concrets#OIC#oi/oic.html*
*3#Environnement Physique#Envir. Physique#/oi/oi/enviphys.html*
*3#Objets Interactifs Abstraits#OIA#/oi/oi/oia.html*
*4#Description des OIA#Des. des OIA#/oi/oi/desoia.html*
*4#Catégories d'OIA#Cat. OIA#/oi/oi/catoia.html*
*3#Transformation d'OIA en OIC#De OIA à OIC#/oi/oi/transoia.html*
*3#Application de Manipulation#Manipulation#/oi/seloia/apps-manip.html*
*3#Diagramme EA#Diagramme EA#/oi/oi/oiea.html*
*2#Base des OIA#Base des OIA#/oi/oia/oia.html*
*3#Classement Alphabétique#Clas. Alphabétique#/oi/oia/alphabétique.html*
*3#Classement par Catégorie#Clas. par Cat.#/oi/oia/categories.html*
*2#Les Arbres de Sélection#Arbres de Sélection#/oi/trees/probleme.html*
*3#Définition d'Arbre#Définition d'Arbre#/oi/trees/def.html*
*3#Critères de sélection#Critères#/oi/trees/criteres.html*
*4#Type d'interaction#Interaction#/oi/trees/typeinter.html*
*4#Type d'information#Type Info#/oi/trees/type.html*
*4#Domaine de définition#Domaine#/oi/trees/domaine.html*
*4#Nombre de choix possibles# Choix Possibles#/oi/trees/choixpos.html*
*4#Nombre de valeurs possibles# Valeurs Possibles#/oi/trees/valpos.html*
*4#Continuité du domaine# Continuité#/oi/trees/continuite.html*
*4#Longueur de l'information# Longueur Info#/oi/trees/longueur.html*
*4#Précision# Précision#/oi/trees/precision.html*
*4#Préférence pour la sélection#Pref. Sélection#/oi/trees/prefsel.html*
*4#Antagonisme#Antagonisme#/oi/trees/antagonisme.html*
*4#Densité d'écran#Densité#/oi/trees/densite.html*
*4#Orientation#Orientation#/oi/trees/orientation.html*
*4#Type de donnée#Type de Donnée#/oi/trees/typedonnee.html*
*4#Nombre de type de donnée#Nbre Type de Donnée#/oi/trees/nbretype.html*
*4#Nombre de valeurs à afficher#Nbre Valeurs af-
ficher#/oi/trees/nbreval.html*
*2#Sélection des OIA#Sélection des OIA#/oi/seloia/probleme.html*
*3#Pour une fenêtre logique#FL#/oi/seloia/selfl.html*
*3#Pour une fonction#Fonction#/oi/seloia/selfonction.html*
*3#Pour un message#Message#/oi/seloia/selmessage.html*
*3#Pour un groupe d'informations#Groupe Info#/oi/seloia/selgroupe.html*
*3#Pour une liste d'informations#Liste Info#/oi/seloia/selliste.html*
*3#Pour une information simple#Info Simple#/oi/seloia/selinfo.html*
*4#Application de Consultation#Consultation#/oi/seloia/apps-consult.html*F
```

F

Interactive Applications

This section contains documents and code related to the interactive applications.

1. IOs Manipulation application

1.1. AbstractCIO class

```
package Vesale.Manipulation;

import javax.swing.*;
import java.awt.Color;

public class AbstractCIO extends JPanel
{
  /* the following methods do nothing */
  /* they will have to be overidded */
  public void setValue(String s) {};
  public String getValue() { return null;};
  public void setLabel(String s) {};
  public String getLabel() { return null;};
  public void setStringValues(String[] newValues){};
  public String[] getStringValues (){ return null;}
  public void setSelectedIconValue (ImageIcon newSelectedValue) {}
  public ImageIcon getSelectedIconValue (){return null;}
  public void setSelectedStringValues (String[] newSelectedValues){}
  public String[] getSelectedStringValues (){ return null;}
  public void setSelectedValues (String[] newSelectedValues){}
  public String[] getSelectedValues (){ return null;}
  public void setSelectedStringValue (String newSelectedValue) {};
  public String getSelectedStringValue (){ return null;}
  public void setSelectedValue (String newSelectedValue) {};
  public String getSelectedValue (){ return null;}
  public void setVisibleRows(int newVisibleRows){}
  public int getVisibleRows() { return -1;}
  public void setAlphabeticalOrder(int newAlphabeticalOrder){}
  public int getAlphabeticalOrder() {     return -1;}
  public void setType(int newType){}
  public int getType() { return -1;}
  public void setSelectedIndex (int newSelectedIndex){}
  public int getSelectedIndex (){   return -1; }
  public void setSelectedIndices (int[] newSelectedIndices){}
  public int[] getSelectedIndices () { return null;}
  public  void  setAllowsMultipleSelection  (boolean  newAllowsMultipleSelec-
tion) {}
  public boolean isAllowsMultipleSelection()    { return false; }
  public void setBackgroundColor(Color newBackgroundColor)  {}
  public Color getBackgroundColor() { return null;  }
  public void setSelectionColor(Color newSelectionColor) {  }
  public Color getSelectionColor() { return null;  }
  public void setEnabled(boolean newEnabled)  {}
  public boolean isEnabled() { return false;}
  public void setLabelPosition(int newLabelPosition) {}
  public int getLabelPosition() { return -1; }
  public void setLabelCentered(boolean newLabelCentered) {}
  public boolean getLabelCentered() { return false; }
  public void setMnemonic(char newMnemonic) {}
  public char getMnemonic() { return ' '; }
  public void setColumns(int newColumns) {}
  public int getColumns() { return -1; }
  public void setValuesLabel(String newValuesLabel) {}
```

```
  public String getValuesLabel() { return null;}
  public void setSelectedValuesLabel (String newSelectedValuesLabel){}
  public String getSelectedValuesLabel() { return null;}
  public void setValuesMnemonic(char newValuesMnemonic) {}
  public char getValuesMnemonic() { return ' ';}
  public void setSelectedValuesMnemonic(char newSelectedValuesMnemonic) { }
  public char getSelectedValuesMnemonic() { return ' ';}
  public void setListType(int newListType){}
  public int getListType() { return -1;}
  public void setOrientation(int newOrientation) {}
  public int getOrientation()  { return -1;}
  public void setOrderButtonsDisplayed(boolean newOrderButtonsDisplayed)  {
}
  public boolean isOrderButtonsDisplayed() { return false; }
  public void setEditable(boolean newEditable)  { }
  public boolean isEditable() { return false; }

  /* this methods is used to get the attributes related to a component */
  public int[] getAttributesList() { return null;}
}
```

1.2. ListBoxAccManipulation class

```
package Vesale.Manipulation.CIO;

import UPE.*;
import Vesale.Manipulation.*;

/* this class extends the ListBoxAcc component of the UPE library */
public class ListBoxAccManipulation extends ListBoxAcc
{

  public ListBoxAccManipulation()
  {
    super();
  }

  public int[] getAttributesList ()
  {
    return new int[]
    {
      ManipulationConstants.ORIENTATION,
      ManipulationConstants.VISIBLEROWS,
      ManipulationConstants.STRINGVALUESACC,
      ManipulationConstants.SELECTEDSTRINGVALUES,
      ManipulationConstants.LABELDISPLAYED,
      ManipulationConstants.LABELDISPLAYEDMNEMONIC,
      ManipulationConstants.LABELSELECTED,
      ManipulationConstants.LABELSELECTEDMNEMONIC,
      ManipulationConstants.ORDERBUTTONSDISPLAYED,
      ManipulationConstants.ALPHABETICALORDER,
      ManipulationConstants.BGCOLOR,
      ManipulationConstants.SELCOLOR,
    };
  }
}
```

2. Selection trees application

2.1. Selection tree description syntax

Here is the syntax that is used to describe the selection tree handled by the selection tree application:

```
<tree> = <inputtree> <outputtree>

<inputtree> = <startchar> <inputtreedescription> <endchar>

<outputtree> = <startchar> <outputtreedescription> <endchar>

<startchar> = "D"

<endchar> = "F"

<linestartchar> = "*"

<lineendchar> = "*"

<separator> = "/"

<emptychar> = "-"

<inputtreedescription> = <inputtreedescriptionrule>*

<outputtreedescription> = <outputtreedescriptionrule>*

<inputtreedescriptionrule> = <linestartchar> <aioid> <separator> <typdesc> <separator> <domain> <separator> <nvc> <separator> <npo> <separator> <continuity> <separator> <length> <separator> <precisicion> <separator> <prefsel> <separator> <opposites> <separator> <density> <lineendchar>

<outputtreedescriptionrule> = <linestartchar> <aioid> <separator> <datatype> <separator> <datanumber> <separator> <nvc> <separator> <typedesc> <separator> <length> <separator> <nvd> <separator> <length> <lineendchar>

<datatype> = S | M | <emptyvalue>

<datanumber> = S | M | <emptyvalue>

<nvd> = S | M | <emptyvalue>

<aioid> = chaine de caractère libre

<typedesc> = A | I | B | E | G | H |D
```

<domain> = I | C | M | <emptyvalue>

<nvc> = S | M | <emptyvalue>

<npo> = chaine de caractère libre | <emptyvalue>

<continuity> = V | F | <emptyvalue>

<length> = V | F | <emptyvalue>

<precision> = E | F | <emptyvalue>

<prefsel> = V | F | <emptyvalue>

<opposites> = V | F | <emptyvalue>

<density> = E | F | <emptyvalue>

2.2. Selection tree description

Here is the description of the selection trees:

```
D*EBX/H/I/S/-/-/-/-/-/-/-*
*RBU/H/C/S/[2,3]/F/-/-/-/-/F*
*DLB/H/C/S/[2,3]/F/-/-/-/-/E*
*RBG/H/C/S/[4,MagN]/F/-/-/-/-/F*
*DLB/H/C/S/[4,MagN]/F/-/-/-/-/E*
*LBX/H/C/S/]MagN,Tm]/F/-/-/-/-/F*
*DLB/H/C/S/]MagN,Tm]/F/-/-/-/-/E*
*DSL/H/C/S/]Tm,+inf]/F/-/-/-/-/F*
*DLB/H/C/S/]Tm,+inf]/F/-/-/-/-/E*
*SBA/H/C/S/[2,Bm]/V/-/E/-/-/-*
*SCA/H/C/S/[2,Bm]/V/-/F/-/-/-*
*TSP/H/C/S/]Bm,Tm]/V/-/E/-/-/-*
*SCA/H/C/S/]Bm,Tm]/V/-/F/-/-/-*
*SEB/H/C/S/>Tm/V/-/-/-/-/-*
*RBE/H/M/S/[2,3]/-/-/-/-/-/F*
*DCB/H/M/S/[2,3]/-/-/-/-/-/E*
*BEG/H/M/S/[4,MagN]/-/-/-/-/-/F*
*DCB/H/M/S/[4,MagN]/-/-/-/-/-/E*
*COB/H/M/S/]MagN,Tm]/-/-/-/-/-/F*
*DCB/H/M/S/]MagN,Tm]/-/-/-/-/-/E*
*SCB/H/M/S/]Tm,+inf]/-/-/-/-/-/F*
*DSC/H/M/S/]Tm,+inf]/-/-/-/-/-/E*
*CBX/H/C/M/[2,3]/-/-/-/-/-/-*
*CBG/H/C/M/[4,MagN]/-/-/-/-/-/-*
*NCA/H/C/M/]MagN,Tm]/-/-/-/-/-/F*
*MLB/H/C/M/]MagN,Tm]/-/-/-/-/-/E*
*BLB/H/C/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/H/C/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/H/C/M/]Tm,+inf]/-/-/-/-/-/E*
*ENA/H/I/M/-/-/-/-/-/-/-*
*CBA/H/M/M/[2,3]/-/-/-/-/-/-*
*CAG/H/M/M/[4,MagN]/-/-/-/-/-/-*
*ECA/H/M/M/]MagN,Tm]/-/-/-/-/-/F*
*MSC/H/M/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/H/M/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/H/M/M/]Tm,+inf]/-/-/-/-/-/E*
*EBX/D/I/S/-/-/-/-/-/-/-*
*RBU/D/C/S/[2,3]/F/-/-/-/-/F*
*DLB/D/C/S/[2,3]/F/-/-/-/-/E*
```

```
*RBG/D/C/S/[4,MagN]/F/-/-/-/-/F*
*DLB/D/C/S/[4,MagN]/F/-/-/-/-/E*
*LBX/D/C/S/]MagN,Tm]/F/-/-/-/-/F*
*DLB/D/C/S/]MagN,Tm]/F/-/-/-/-/E*
*DSL/D/C/S/]Tm,+inf]/F/-/-/-/-/F*
*DLB/D/C/S/]Tm,+inf]/F/-/-/-/-/E*
*CAL/D/C/S/[2,Bm]/V/-/-/V/-/F*
*DCA/D/C/S/[2,Bm]/V/-/-/V/-/E*
*MSE/D/C/S/[2,Bm]/V/-/-/F/-/-*
*DSP/D/C/S/]Bm,Tm]/V/-/-/V/-/-*
*MSE/D/C/S/]Bm,Tm]/V/-/-/F/-/-*
*MSE/D/C/S/>Tm/V/-/-/-/-/-*
*RBE/D/M/S/[2,3]/-/-/-/-/-/F*
*DCB/D/M/S/[2,3]/-/-/-/-/-/E*
*BEG/D/M/S/[4,MagN]/-/-/-/-/-/F*
*DCB/D/M/S/[4,MagN]/-/-/-/-/-/E*
*COB/D/M/S/]MagN,Tm]/-/-/-/-/-/F*
*DCB/D/M/S/]MagN,Tm]/-/-/-/-/-/E*
*SCB/D/M/S/]Tm,+inf]/-/-/-/-/-/F*
*DSC/D/M/S/]Tm,+inf]/-/-/-/-/-/E*
*CBX/D/C/M/[2,3]/-/-/-/-/-/-*
*CBG/D/C/M/[4,MagN]/-/-/-/-/-/-*
*NCA/D/C/M/]MagN,Tm]/-/-/-/-/-/F*
*MLB/D/C/M/]MagN,Tm]/-/-/-/-/-/E*
*BLB/D/C/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/D/C/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/D/C/M/]Tm,+inf]/-/-/-/-/-/E*
*ENA/D/I/M/-/-/-/-/-/-/-*
*CBA/D/M/M/[2,3]/-/-/-/-/-/-*
*CAG/D/M/M/[4,MagN]/-/-/-/-/-/-*
*ECA/D/M/M/]MagN,Tm]/-/-/-/-/-/F*
*MSC/D/M/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/D/M/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/D/M/M/]Tm,+inf]/-/-/-/-/-/E*
*VSW/B/-/-/-/-/-/-/-/V/-*
*CBX/B/-/-/-/-/-/-/-/F/-*
*EGW/G/I/S/-/-/-/-/-/-/F*
*BBU/G/I/S/-/-/-/-/-/-/E*
*VRI/G/C/S/[2,3]/-/-/-/-/-/F*
*DGL/G/C/S/[2,3]/-/-/-/-/-/E*
*VRG/G/C/S/[4,MagN]/-/-/-/-/-/F*
*DGL/G/C/S/[4,MagN]/-/-/-/-/-/E*
*LIG/G/C/S/]MagN,Tm]/-/-/-/-/-/F*
*DGL/G/C/S/]MagN,Tm]/-/-/-/-/-/E*
*SGB/G/C/S/]Tm,+inf]/-/-/-/-/-/F*
*SDG/G/C/S/]Tm,+inf]/-/-/-/-/-/E*
*RIW/G/M/S/[2,3]/-/-/-/-/-/F*
*RIB/G/M/S/[2,3]/-/-/-/-/-/E*
*RWG/G/M/S/[4,MagN]/-/-/-/-/-/F*
*IBB/G/M/S/[4,MagN]/-/-/-/-/-/E*
*GCB/G/M/S/]MagN,Tm]/-/-/-/-/-/F*
*DGC/G/M/S/]MagN,Tm]/-/-/-/-/-/E*
*SGR/G/M/S/]Tm,+inf]/-/-/-/-/-/F*
*SGC/G/M/S/]Tm,+inf]/-/-/-/-/-/E*
*BBA/G/I/M/-/-/-/-/-/-/-*
*CBI/G/C/M/[2,3]/-/-/-/-/-/-*
*CIG/G/C/M/[4,MagN]/-/-/-/-/-/-*
*LIG/G/C/M/]MagN,Tm]/-/-/-/-/-/F*
*MGL/G/C/M/]MagN,Tm]/-/-/-/-/-/E*
*NGA/G/C/M/]MagN,Tm]/-/-/-/-/-/E*
*LIG/G/C/M/]Tm,+inf]/-/-/-/-/-/F*
*SGL/G/C/M/]Tm,+inf]/-/-/-/-/-/E*
*SNG/G/C/M/]Tm,+inf]/-/-/-/-/-/E*
*CIW/G/M/M/[2,3]/-/-/-/-/-/-*
*CEW/G/M/M/[4,MagN]/-/-/-/-/-/-*
*EGA/G/M/M/]MagN,Tm]/-/-/-/-/-/-*
*SGA/G/M/M/]Tm,+inf]/-/-/-/-/-/-*
*EBX/I/I/S/-/-/-/-/-/-/-*
*RBU/I/C/S/[2,3]/F/-/-/-/-/F*
*DLB/I/C/S/[2,3]/F/-/-/-/-/E*
*RBG/I/C/S/[4,MagN]/F/-/-/-/-/F*
*DLB/I/C/S/[4,MagN]/F/-/-/-/-/E*
*LBX/I/C/S/]MagN,Tm]/F/-/-/-/-/F*
*DLB/I/C/S/]MagN,Tm]/F/-/-/-/-/E*
*DSL/I/C/S/]Tm,+inf]/F/-/-/-/-/F*
*DLB/I/C/S/]Tm,+inf]/F/-/-/-/-/E*
*VTH/I/C/S/[2,Bm]/V/-/E/-/-/-*
```

```
*SCA/I/C/S/[2,Bm]/V/-/F/-/-/-*
*SPB/I/C/S/]Bm,Tm]/V/-/E/-/-/-*
*VSB/I/C/S/]Bm,Tm]/V/-/F/-/-/-*
*SSP/I/C/S/>Tm/V/-/-/-/-/-*
*RBE/I/M/S/[2,3]/-/-/-/-/-/F*
*DCB/I/M/S/[2,3]/-/-/-/-/-/E*
*BEG/I/M/S/[4,MagN]/-/-/-/-/-/F*
*DCB/I/M/S/[4,MagN]/-/-/-/-/-/E*
*COB/I/M/S/]MagN,Tm]/-/-/-/-/-/F*
*DCB/I/M/S/]MagN,Tm]/-/-/-/-/-/E*
*SCB/I/M/S/]Tm,+inf]/-/-/-/-/-/F*
*DSC/I/M/S/]Tm,+inf]/-/-/-/-/-/E*
*CBX/I/C/M/[2,3]/-/-/-/-/-/-*
*CBG/I/C/M/[4,MagN]/-/-/-/-/-/-*
*NCA/I/C/M/]MagN,Tm]/-/-/-/-/-/F*
*MLB/I/C/M/]MagN,Tm]/-/-/-/-/-/E*
*BLB/I/C/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/I/C/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/I/C/M/]Tm,+inf]/-/-/-/-/-/E*
*ENA/I/I/M/-/-/-/-/-/-/-*
*CBA/I/M/M/[2,3]/-/-/-/-/-/-*
*CAG/I/M/M/[4,MagN]/-/-/-/-/-/-*
*ECA/I/M/M/]MagN,Tm]/-/-/-/-/-/F*
*MSC/I/M/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/I/M/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/I/M/M/]Tm,+inf]/-/-/-/-/-/E*
*EBX/R/I/S/-/-/-/-/-/-/-*
*RBU/R/C/S/[2,3]/F/-/-/-/-/F*
*DLB/R/C/S/[2,3]/F/-/-/-/-/E*
*RBG/R/C/S/[4,MagN]/F/-/-/-/-/F*
*DLB/R/C/S/[4,MagN]/F/-/-/-/-/E*
*LBX/R/C/S/]MagN,Tm]/F/-/-/-/-/F*
*DLB/R/C/S/]MagN,Tm]/F/-/-/-/-/E*
*DSL/R/C/S/]Tm,+inf]/F/-/-/-/-/F*
*DLB/R/C/S/]Tm,+inf]/F/-/-/-/-/E*
*RBE/R/M/S/[2,3]/-/-/-/-/-/F*
*DCB/R/M/S/[2,3]/-/-/-/-/-/E*
*BEG/R/M/S/[4,MagN]/-/-/-/-/-/F*
*DCB/R/M/S/[4,MagN]/-/-/-/-/-/E*
*COB/R/M/S/]MagN,Tm]/-/-/-/-/-/F*
*DCB/R/M/S/]MagN,Tm]/-/-/-/-/-/E*
*SCB/R/M/S/]Tm,+inf]/-/-/-/-/-/F*
*DSC/R/M/S/]Tm,+inf]/-/-/-/-/-/E*
*CBX/R/C/M/[2,3]/-/-/-/-/-/-*
*CBG/R/C/M/[4,MagN]/-/-/-/-/-/-*
*NCA/R/C/M/]MagN,Tm]/-/-/-/-/-/F*
*MLB/R/C/M/]MagN,Tm]/-/-/-/-/-/E*
*BLB/R/C/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/R/C/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/R/C/M/]Tm,+inf]/-/-/-/-/-/E*
*ENA/R/I/M/-/-/-/-/-/-/-*
*CBA/R/M/M/[2,3]/-/-/-/-/-/-*
*CAG/R/M/M/[4,MagN]/-/-/-/-/-/-*
*ECA/R/M/M/]MagN,Tm]/-/-/-/-/-/F*
*MSC/R/M/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/R/M/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/R/M/M/]Tm,+inf]/-/-/-/-/-/E*
*EBX/A/I/S/-/-/F/-/-/-/-*
*MLE/A/I/S/-/-/V/-/-/-/-*
*ENA/A/I/M/-/-/F/-/-/-/-*
*AME/A/I/M/-/-/V/-/-/-/-*
*RBU/A/C/S/[2,3]/F/-/-/-/-/F*
*DLB/A/C/S/[2,3]/F/-/-/-/-/E*
*RBG/A/C/S/[4,MagN]/F/-/-/-/-/F*
*DLB/A/C/S/[4,MagN]/F/-/-/-/-/E*
*LBX/A/C/S/]MagN,Tm]/F/-/-/-/-/F*
*DLB/A/C/S/]MagN,Tm]/F/-/-/-/-/E*
*DSL/A/C/S/]Tm,+inf]/F/-/-/-/-/F*
*DLB/A/C/S/]Tm,+inf]/F/-/-/-/-/E*
*RBE/A/M/S/[2,3]/-/-/-/-/-/F*
*DCB/A/M/S/[2,3]/-/-/-/-/-/E*
*BEG/A/M/S/[4,MagN]/-/-/-/-/-/F*
*DCB/A/M/S/[4,MagN]/-/-/-/-/-/E*
*COB/A/M/S/]MagN,Tm]/-/-/-/-/-/F*
*DCB/A/M/S/]MagN,Tm]/-/-/-/-/-/E*
*SCB/A/M/S/]Tm,+inf]/-/-/-/-/-/F*
*DSC/A/M/S/]Tm,+inf]/-/-/-/-/-/E*
```

```
*CBX/A/C/M/[2,3]/-/-/-/-/-/-*
*CBG/A/C/M/[4,MagN]/-/-/-/-/-/-*
*NCA/A/C/M/]MagN,Tm]/-/-/-/-/-/F*
*MLB/A/C/M/]MagN,Tm]/-/-/-/-/-/E*
*BLB/A/C/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/A/C/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/A/C/M/]Tm,+inf]/-/-/-/-/-/E*
*ENA/A/I/M/-/-/-/-/-/-/-*
*CBA/A/M/M/[2,3]/-/-/-/-/-/-*
*CAG/A/M/M/[4,MagN]/-/-/-/-/-/-*
*ECA/A/M/M/]MagN,Tm]/-/-/-/-/-/F*
*MSC/A/M/M/]MagN,Tm]/-/-/-/-/-/E*
*SEA/A/M/M/]Tm,+inf]/-/-/-/-/-/F*
*SSC/A/M/M/]Tm,+inf]/-/-/-/-/-/E*F
D*MSE/S/-/H/-/S/-*
*LBX/S/-/H/-/M/-*
*MSE/S/-/D/-/S/-*
*LBX/S/-/D/-/M/-*
*SIC/S/-/G/-/S/-*
*PLB/S/-/G/-/M/-*
*CBX/S/-/B/-/S/-*
*BLB/S/-/B/-/M/-*
*MSE/S/-/I/-/S/-*
*LBX/S/-/I/-/M/-*
*MSE/S/-/R/-/S/-*
*LBX/S/-/R/-/M/-*
*MSE/S/-/A/-/S/F*
*KEB/S/-/A/-/S/V*
*LBX/S/-/A/-/M/-*
*EXT/M/M/-/F/-/-*
*PAN/M/M/-/V/-/-*
*TME/M/S/H/-/-/-*
*TME/M/S/D/-/-/-*
*TIC/M/S/G/-/-/-*
*BLB/M/S/B/-/-/-*
*TME/M/S/I/-/-/-*
*TME/M/S/R/-/-/-*
*TEB/M/S/A/-/-/F*
*TLE/M/S/A/-/-/V*F
```

2.3. Selection tree class

```
package Vesale.TreeExplorer;

import java.io.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;
import java.lang.*;
import UPE.*;
import java.net.*;

public class SelectionTree
{

/* constructor */
public SelectionTree ()
{
  /* read the selection tree from the text file */
  parseTree();
  /* read information about AIOs from the text file */
  parseAIOs();
}


/* Input and Out trees */
```

```
private Vector inputtree = new Vector();
private Vector outputtree = new Vector();

private Hashtable aiosIdKey = new Hashtable();
private Hashtable aiosNameKey = new Hashtable();

/* constants used to parse the tree */
protected static int STARTLETTER = 'D';
protected static int ENDLETTER = 'F';
protected static int FIELDSEPARATOR = '/';
protected static int FIELDSEPARATORAIO = '#';
protected static char EMPTYCHAR = '-';
protected static char RULESTART = '*';

String fileContent;
int pos = 0;
char c;

/* Public methods */

/**
* A partir d'un OIA, d'un type d'interaction, d'un type de valeur et d'une
* liste de couples (critères,valeurs), renvoie une liste de critval
*nécessaire pour que cet objet soit sélectionnable */
public Vector getRedAIOsChanges(String aio, int interaction,
              String type, Vector critvalvector)
{
  if (interaction == ExplorerConstants.INPUT)
  {
    return getRedAIOsChangesInput(aio,type,critvalvector);
  }
  else
  {
    return getRedAIOsChangesOutput(aio,critvalvector);
  }
}



/**
* A partir d'un OIA, d'un type d'interaction, d'un type de valeur et d'une
* liste
* de couples (critères,valeurs), renvoie un vector de vector de critval
* nécessaire pour que cet objet soit le seul sélectionné */
public Vector getGreenAIOsChanges(String aio, int interaction,
              String type, Vector critvalvector)
{
  if (interaction == ExplorerConstants.INPUT)
  {
    return getGreenAIOsChangesInput(aio,type,critvalvector);
  }
  else
  {
    return getGreenAIOsChangesOutput(aio,critvalvector);
  }
}

/**
* A partir d'un type d'intercation, d'un type de valeur et d'une liste de
* couples (critères,valeurs), renvoie la liste des OIA correspondants
**/
```

```
public AIOsPartition getAIOs (int interaction, String type,
                        Vector critvalvector)
{
  if (interaction == ExplorerConstants.INPUT)
  {
    return getAIOsInput(type,critvalvector);
  }
  else
  {
    return getAIOsOutput(critvalvector);
  }
}

/**
* A partir d'un type d'interaction et d'un type, renvoie les critères
* intervenant dans cet arbre
**/
public Vector getCriteria (int interaction, String type)
{
try
{
  if (interaction == ExplorerConstants.INPUT)
  {
    return getCriteriaInput(type);
  }
  else
  {
    return getCriteriaOutput();
  }
}
catch (Exception e) { return new Vector();}
}

/**
* A partir d'un critères, renvoie les différentes valeurs présentes
* dans un certain arbre
**/
public Vector getCriterionValues (int interaction, String type, int crite-
rion)
{
try
{
  if (interaction == ExplorerConstants.INPUT)
  {
    return getCriterionValuesInput(type,criterion);
  }
  else
  {
    return getCriterionValuesOutput(criterion);
  }
}
catch (Exception e) { return new Vector(); }
}

/* retourne le nom de l'oia */
public String getAIOName(String id)
{
  try
  {
    return ((AIO)aiosIdKey.get(id)).getName();
  }
```

```
    catch (Exception e) { return "connait pas";}
}

/* retourne le nom de l'oia */
public boolean getAIOManip(String id)
{
    return ((AIO)aiosIdKey.get(id)).isManip();
}

/* retourne le nom de l'oia */
public String getAIOPic(String id)
{
    return ((AIO)aiosIdKey.get(id)).getPic();
}


/* retourne l'id de l'oia */
public String getAIOId(String name)
{
    return (String)aiosNameKey.get(name);
}

/* retourne le tableau de noms d'oia */
public String[] getAIOsNames(String[] tab)
{
    String[] newTab = new String[tab.length];
    for (int i = 0 ; i < tab.length; i++)
    {
      newTab[i] = getAIOName(tab[i]);
    }
    return newTab;
}

/* getAIOsIds */
public String[] getAIOsIds(String[] tab)
{
    String[] newTab = new String[tab.length];
    for (int i = 0 ; i < tab.length; i++)
    {
      newTab[i] = getAIOId(tab[i]);
    }
    return newTab;
}

/* Private methods */
// if your are interested to this private methods, you can contact us //
}
```

3. Justification of the AIOs selected

In order to justify the selection of the AIOs used in the applications, we will use Jean Vanderdonckt's selection trees[1].

3.1. IOs Manipulation application

3.1.1. AIO zone

The required AIO is:

- **AIO for the selection of the manipulated AIO**

Criteria:

- Interaction type = Input/Output
- Value type = Alphanumeric
- Number of values to choose = 1
- Known Domain (the information concerning the AIOs that can be manipulated is stored in the IOs database)
- Number of possibles options in]MagN,Tm] (depends on the number of AIOs that can be manipulated)
- Screen density = High

Selected AIO = Drop-Down List Box

3.1.2. Attributes zone

The required AIO are:

- **AIO for the assignment/displaying of a non-composed value**

Criteria:

- Interaction type = Input/Output
- Value type = Alphanumeric
- Number of values to choose = 1
- Known Domain
- Number of possibles options in]MagN,Tm] (depends on the attribute)
- Screen density = High

Selected AIO = Drop-Down List Box

[1] See appendix A section 8.1

- **AIO for the displaying of a composed value**

Criteria:

- Interaction type = Output
- Data type = Elementary
- Value type = Alphanumeric
- Number of values to display = 1
- Length > Lm

Selected AIO = Multi-Line Edit Box (non editable)

3.2. Selection trees application

3.2.1. Criteria Zone

The required AIOs are:

- **AIO for the selection of the value type**

Criteria:

- Interaction type = Input/Output
- Value type = Alphanumeric
- Number of values to choose = 1
- Known Domain
- Number of possibles options in]MagN,Tm]
- Screen density = High

Selected AIO = Drop-Down List Box

- **AIO for the selection of a criterion value**

We would like to select the same AIO for all the criteria in order to improve the homogeneity of the application.

The criteria have the same values as for the value type AIO (see above). Therefore, the AIOs used will be Drop-Down List Boxes.

- **AIO to enable/disable the criterion**

Criteria:

- Interaction type = Input/Output
- Value type = Boolean because the state of the criterion is either disabled or enabled.
- Opposites values = false

Selected AIO = Check Box

3.2.2. Red / Green Zone

The required AIOs are:

- **AIO used to display the AIOs that can be selected**

Criteria:

- Interaction type = Input/Output
- Value type = Alphanumeric
- Number of values to choose = 1
- Known Domain
- Number of possibles options in]MagN,Tm]
- Screen density = Low because only two AIOs will be displayed in that zone

Selected AIO = List Box

- **AIO used to display the AIOs that can't be selected**

See the selection of an AIO to display the AIOs that can be selected.

3.2.3. Advice Zone

- **AIO used to display an advice**

Criteria:

- Interaction type = Output
- Data type = Elementary
- Value type = Alphanumeric
- Number of values to display = 1
- Length > Lm

Selected AIO = Multi-Line Edit Box (non editable)

- **AIO used to execute the advice automatically**

To launch an action, the selected AIO is a button.

3.2.4. AIO Zone

The required AIOs are:

- **AIO used to display the AIO picture**

Criteria:

- Interaction type = Output
- Data type = Elementary
- Value type = graphical

- Number of values to display = 1

Selected AIO = Static Icon